

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Serial No. 10/607,054
Filing Date 06/25/2003
Inventorship Prabu et al.
Applicant Microsoft Corp.
Group Art Unit 2166
Examiner Lin, Shew Fen
Attorney's Docket No. 302701.01
Title: Using Task Sequences to Manage Devices

**DECLARATION SHOWING REFERENCE'S DISCLOSURE
WAS DERIVED FROM APPLICANT'S OWN WORK**

We hereby declare that we are co-inventors of the subject matter described in the above-identified patent application.

We are aware that a publication entitled, "Image-based Installation of the Operating System and the Cluster Service Using Automated Deployment Services," published on January 1, 2003 and located at the link: <http://technet2.microsoft.com/WindowsServer/en/library/ba62f36-2a9d043d209737-ab50d5b8b71b1033.msp?mfr=true>, has been cited against the above-identified patent application. On information and belief, the website from which this article originated is a website (Microsoft TechNet) that is sponsored by Microsoft Corp., the current assignee of this application. The Microsoft TechNet website describes various products offered by Microsoft Corp..

On information and belief, we hereby declare that the above-mentioned article describes our own work as set forth in the above-identified patent application. To support this declaration, we submit the following facts which are supported by the attached exhibits. In the

1 discussion that follows, various similarities between documents authored
2 by the inventors and the reference cited by the Patent Office are identified.

3 On information and belief, attached as Exhibit 1 is the publication
4 entitled "Image-based Installation of the Operating System and the Cluster
5 Service Using Automated Deployment Services" (hereinafter "ADS").
6 The ADS reference has been cited by the Patent Office.

7 On information and belief, attached as Exhibit 2 is a document
8 entitled "Disclosure Packet" and bearing number 302701.1 entitled
9 "Methodology for Task Sequence Scheduling". On information and
10 belief, this document pertains to the above-identified patent application
11 and was used as part of the scheduling process to schedule a disclosure
12 meeting for the subject matter that is the subject of the above-identified
13 patent application. The document includes, as attachments, four separate
14 documents. Two of these documents are encircled and are entitled
15 respectively, "Task Sequence Spec" and "ADS Functional Overview".
16 Some similarities between these attached documents and Exhibit 1 are
17 discussed below.

18 On information and belief, attached as Exhibit 3 is a document
19 entitled "Task Sequences Functional Specification". This document
20 corresponds to the "Task Sequence Spec" document attached to the
21 "Disclosure Packet" (Exhibit 2).

22 On information and belief, attached as Exhibit 4 is a document
23 entitled "ADS Functional Overview". This document corresponds to the
24 "ADS Functional Overview" document attached to the "Disclosure
25 Packet" (Exhibit 2).

The "ADS Functional Overview" Document

Starting first with the document entitled "ADS Functional Overview" (Exhibit 4), the cover page of this document indicates a copyright date of 2001 (first bracket). Three of the present inventors are listed respectively on the cover page as "PM Author", "Dev Author", and "Test Contact" (second bracket).

On page 3 of Exhibit 4, this document states that "Microsoft is adding an enhancement to the .NET server platform called ADS." (first bracket). Additionally, on this page, and bracketed by the second bracket designated "A", appears a discussion of two generic operations capabilities that ADS will provide. These capabilities are likewise mentioned in Exhibit 1 and are set off by the bracket designated "A" on page 1.

On page 13 of Exhibit 4, and set off by the bracket designated "B", appears a discussion of imaging support provided by ADS. Specifically, this excerpt of text describes capturing and deploying images using "sysprep", which is part of the Windows Server OPK. On the third page of Exhibit 1, and appearing bracketed by a bracket designated "B" appears a discussion of how to create a master image. Notice in the discussion that the master installation is prepared with the "Sysprep" tool.

On page 13 of Exhibit 4, and set off by the bracket designated "C" is a discussion entitled "Task Sequences". This discussion indicates that a task sequence is a sequence of operations to be performed in order. The sequence definition is stored in an XML file on the controller. On page 8 of Exhibit 1 under the heading "Create task sequence file for image

1 employment" and bracketed by a bracket designated "C", appears a
2 discussion of how to create a task sequence file for the ADS controller.
3 This discussion describes the notion of creating a task sequence which is
4 an XML file containing a sequence of tasks for a controller to perform.

5
6 **The "Task Sequences Functional Specification" Document**

7 Turning attention to the "Task Sequences Functional Specification"
8 document (Exhibit 3), such indicates on its cover page that three of the
9 inventors were developer authors (indicated in the bracket).

10 On page 3 under the table of contents, a section designated 2.1.2.2
11 is entitled "Task sequences and Workflow Overview". In addition, a
12 section on page 4 designated 2.1.7 is entitled "Task sequence
13 Implementation". These sections correspond in content to content that
14 appears in Exhibit 1 beginning on page 8 under the heading "Create a job
15 template for the sequence file".

16 On page 6 of Exhibit 3 within the bracket designated "C" appears a
17 discussion pertaining to a task sequence. Such corresponds in content to
18 the discussion on page 8 of Exhibit 1.

19 On page 21 of Exhibit 3 appears a discussion the XML schema
20 including, at item "D", the task element which includes child elements
21 including the command and parameters element (item "E"). This
22 corresponds in content with the discussion on page 9 of Exhibit 1 which
23 illustrates a sample XML excerpt that includes the task and parameters
24 elements designated respectively, at "D" and "E". A discussion of the
25 parameters element occurs on page 22 of Exhibit 3.

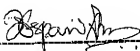
1 On page 11 of Exhibit 3 appears a discussion, at item "F" of a job
2 template that can be created that refers to the task sequence. This
3 corresponds in content with the discussion on page 9 of Exhibit 1 at item
4 "F" entitled "Create a job template for the sequence file".

5 Based upon the similarities between Exhibits 3 and 4, which were
6 authored by subsets of the inventors, and Exhibit 1 - the ADS reference
7 cited by the Patent Office, as well as other similarities which are not
8 specifically identified above, it should be apparent that Exhibit 1 was
9 derived from and describes the work of the inventors as set forth in
10 Exhibits 3 and 4.

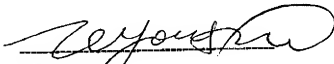
11 We further declare that all statements made herein of our own
12 knowledge are true and that all statements made on information and belief
13 are believed to be true; and further that these statements were made with
14 the knowledge that willful false statements and the like so made are
15 punishable by fine or imprisonment, or both, under 18 U.S.C. 1001, and
16 that such willful false statements may jeopardize the validity of the
17 application or any patent issued thereon.

18 
19 Curt A. Steeb

20 Co-Inventor of Application Serial No. 10/607,054
21
22
23
24
25



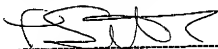
Munisamy Prabu
Co-Inventor of Application Serial No. 10/607,054



Zeyong Xu
Co-Inventor of Application Serial No. 10/607,054




Martin Holladay
Co-Inventor of Application Serial No. 10/607,054



Paul Sutton
Co-Inventor of Application Serial No. 10/607,054


Raymond Pedrizetti

Co-Inventor of Application Serial No. 10/607,054


Michael Gallop

Co-Inventor of Application Serial No. 10/607,054

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Serial No.10/607,054
Filing Date06/25/2003
InventorshipPrabu et al.
ApplicantMicrosoft Corp.
Group Art Unit2166
ExaminerLin, Shew Fen
Attorney's Docket No.302701.01
Title: Using Task Sequences to Manage Devices

**DECLARATION SHOWING REFERENCE'S DISCLOSURE
WAS DERIVED FROM APPLICANT'S OWN WORK**

We hereby declare that we are co-inventors of the subject matter described in the above-identified patent application.

We are aware that a publication entitled, "Image-based Installation of the Operating System and the Cluster Service Using Automated Deployment Services," published on January 1, 2003 and located at the link: <http://technet2.microsoft.com/WindowsServer/en/library/ba62f36-2a9d043d209737-ab50d5b8b71b1033.msp?mfr=true>, has been cited against the above-identified patent application. On information and belief, the website from which this article originated is a website (Microsoft TechNet) that is sponsored by Microsoft Corp., the current assignee of this application. The Microsoft TechNet website describes various products offered by Microsoft Corp..

On information and belief, we hereby declare that the above-mentioned article describes our own work as set forth in the above-identified patent application. To support this declaration, we submit the following facts which are supported by the attached exhibits. In the

1 discussion that follows, various similarities between documents authored
2 by the inventors and the reference cited by the Patent Office are identified.

3 On information and belief, attached as Exhibit 1 is the publication
4 entitled "Image-based Installation of the Operating System and the Cluster
5 Service Using Automated Deployment Services" (hereinafter "ADS").
6 The ADS reference has been cited by the Patent Office.

7 On information and belief, attached as Exhibit 2 is a document
8 entitled "Disclosure Packet" and bearing number 302701.1 entitled
9 "Methodology for Task Sequence Scheduling". On information and
10 belief, this document pertains to the above-identified patent application
11 and was used as part of the scheduling process to schedule a disclosure
12 meeting for the subject matter that is the subject of the above-identified
13 patent application. The document includes, as attachments, four separate
14 documents. Two of these documents are encircled and are entitled
15 respectively, "Task Sequence Spec" and "ADS Functional Overview".
16 Some similarities between these attached documents and Exhibit 1 are
17 discussed below.

18 On information and belief, attached as Exhibit 3 is a document
19 entitled "Task Sequences Functional Specification". This document
20 corresponds to the "Task Sequence Spec" document attached to the
21 "Disclosure Packet" (Exhibit 2).

22 On information and belief, attached as Exhibit 4 is a document
23 entitled "ADS Functional Overview". This document corresponds to the
24 "ADS Functional Overview" document attached to the "Disclosure
25 Packet" (Exhibit 2).

1
2 **The “ADS Functional Overview” Document**

3 Starting first with the document entitled “ADS Functional
4 Overview” (Exhibit 4), the cover page of this document indicates a
5 copyright date of 2001 (first bracket). Three of the present inventors are
6 listed respectively on the cover page as “PM Author”, “Dev Author”, and
7 “Test Contact” (second bracket).

8 On page 3 of Exhibit 4, this document states that “Microsoft is
9 adding an enhancement to the .NET server platform called ADS.” (first
10 bracket). Additionally, on this page, and bracketed by the second bracket
11 designated “A”, appears a discussion of two generic operations capabilities
12 that ADS will provide. These capabilities are likewise mentioned in
13 Exhibit 1 and are set off by the bracket designated “A” on page 1.

14 On page 13 of Exhibit 4, and set off by the bracket designated “B”,
15 appears a discussion of imaging support provided by ADS. Specifically,
16 this excerpt of text describes capturing and deploying images using
17 “sysprep”, which is part of the Windows Server OPK. On the third page
18 of Exhibit 1, and appearing bracketed by a bracket designated “B” appears
19 a discussion of how to create a master image. Notice in the discussion that
20 the master installation is prepared with the “Sysprep” tool.

21 On page 13 of Exhibit 4, and set off by the bracket designated “C”
22 is a discussion entitled “Task Sequences”. This discussion indicates that a
23 task sequence is a sequence of operations to be performed in order. The
24 sequence definition is stored in an XML file on the controller. On page 8
25 of Exhibit 1 under the heading “Create task sequence file for image

1 employment” and bracketed by a bracket designated “C”, appears a
2 discussion of how to create a task sequence file for the ADS controller.
3 This discussion describes the notion of creating a task sequence which is
4 an XML file containing a sequence of tasks for a controller to perform.

5 6 **The “Task Sequences Functional Specification” Document**

7 Turning attention to the “Task Sequences Functional Specification”
8 document (Exhibit 3), such indicates on its cover page that three of the
9 inventors were developer authors (indicated in the bracket).

10 On page 3 under the table of contents, a section designated 2.1.2.2
11 is entitled “Task sequences and Workflow Overview”. In addition, a
12 section on page 4 designated 2.1.7 is entitled “Task sequence
13 Implementation”. These sections correspond in content to content that
14 appears in Exhibit 1 beginning on page 8 under the heading “Create a job
15 template for the sequence file”.

16 On page 6 of Exhibit 3 within the bracket designated “C” appears a
17 discussion pertaining to a task sequence. Such corresponds in content to
18 the discussion on page 8 of Exhibit 1.

19 On page 21 of Exhibit 3 appears a discussion the XML schema
20 including, at item “D”, the task element which includes child elements
21 including the command and parameters element (item “E”). This
22 corresponds in content with the discussion on page 9 of Exhibit 1 which
23 illustrates a sample XML excerpt that includes the task and parameters
24 elements designated respectively, at “D” and “E”. A discussion of the
25 parameters element occurs on page 22 of Exhibit 3.

1 On page 11 of Exhibit 3 appears a discussion, at item "F" of a job
2 template that can be created that refers to the task sequence. This
3 corresponds in content with the discussion on page 9 of Exhibit 1 at item
4 "F" entitled "Create a job template for the sequence file".

5 Based upon the similarities between Exhibits 3 and 4, which were
6 authored by subsets of the inventors, and Exhibit 1 – the ADS reference
7 cited by the Patent Office, as well as other similarities which are not
8 specifically identified above, it should be apparent that Exhibit 1 was
9 derived from and describes the work of the inventors as set forth in
10 Exhibits 3 and 4.

11 We further declare that all statements made herein of our own
12 knowledge are true and that all statements made on information and belief
13 are believed to be true; and further that these statements were made with
14 the knowledge that willful false statements and the like so made are
15 punishable by fine or imprisonment, or both, under 18 U.S.C. 1001, and
16 that such willful false statements may jeopardize the validity of the
17 application or any patent issued thereon.

18
19 _____
20 Curt A. Steeb
21 Co-Inventor of Application Serial No. 10/607,054
22
23
24
25

1
2 Munisamy Prabu
3 Co-Inventor of Application Serial No. 10/607,054
4

5 Zeyong Xu
6 Co-Inventor of Application Serial No. 10/607,054
7

8 Martin Holladay
9 Co-Inventor of Application Serial No. 10/607,054
10

11 Paul Sutton
12 Co-Inventor of Application Serial No. 10/607,054
13

14 
15 Raymond Pedrizetti
16 Co-Inventor of Application Serial No. 10/607,054
17

18 Michael Gallop
19 Co-Inventor of Application Serial No. 10/607,054
20
21
22
23
24
25

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Serial No. 10/607,054
Filing Date 06/25/2003
Inventorship Prabu et al.
Applicant Microsoft Corp.
Group Art Unit 2166
Examiner Lin, Shew Fen
Attorney's Docket No. 302701.01
Title: Using Task Sequences to Manage Devices

**DECLARATION SHOWING REFERENCE'S DISCLOSURE
WAS DERIVED FROM APPLICANT'S OWN WORK**

We hereby declare that we are co-inventors of the subject matter described in the above-identified patent application.

We are aware that a publication entitled, "Image-based Installation of the Operating System and the Cluster Service Using Automated Deployment Services," published on January 1, 2003 and located at the link: <http://technet2.microsoft.com/WindowsServer/en/library/ba62f36-2a9d043d209737-ab50d5b8b71b1033.msp?mfr=true>, has been cited against the above-identified patent application. On information and belief, the website from which this article originated is a website (Microsoft TechNet) that is sponsored by Microsoft Corp., the current assignee of this application. The Microsoft TechNet website describes various products offered by Microsoft Corp..

On information and belief, we hereby declare that the above-mentioned article describes our own work as set forth in the above-identified patent application. To support this declaration, we submit the following facts which are supported by the attached exhibits. In the

1 discussion that follows, various similarities between documents authored
2 by the inventors and the reference cited by the Patent Office are identified.

3 On information and belief, attached as Exhibit 1 is the publication
4 entitled "Image-based Installation of the Operating System and the Cluster
5 Service Using Automated Deployment Services" (hereinafter "ADS").
6 The ADS reference has been cited by the Patent Office.

7 On information and belief, attached as Exhibit 2 is a document
8 entitled "Disclosure Packet" and bearing number 302701.1 entitled
9 "Methodology for Task Sequence Scheduling". On information and
10 belief, this document pertains to the above-identified patent application
11 and was used as part of the scheduling process to schedule a disclosure
12 meeting for the subject matter that is the subject of the above-identified
13 patent application. The document includes, as attachments, four separate
14 documents. Two of these documents are encircled and are entitled
15 respectively, "Task Sequence Spec" and "ADS Functional Overview".
16 Some similarities between these attached documents and Exhibit 1 are
17 discussed below.

18 On information and belief, attached as Exhibit 3 is a document
19 entitled "Task Sequences Functional Specification". This document
20 corresponds to the "Task Sequence Spec" document attached to the
21 "Disclosure Packet" (Exhibit 2).

22 On information and belief, attached as Exhibit 4 is a document
23 entitled "ADS Functional Overview". This document corresponds to the
24 "ADS Functional Overview" document attached to the "Disclosure
25 Packet" (Exhibit 2).

The "ADS Functional Overview" Document

Starting first with the document entitled "ADS Functional Overview" (Exhibit 4), the cover page of this document indicates a copyright date of 2001 (first bracket). Three of the present inventors are listed respectively on the cover page as "PM Author", "Dev Author", and "Test Contact" (second bracket).

On page 3 of Exhibit 4, this document states that "Microsoft is adding an enhancement to the .NET server platform called ADS." (first bracket). Additionally, on this page, and bracketed by the second bracket designated "A", appears a discussion of two generic operations capabilities that ADS will provide. These capabilities are likewise mentioned in Exhibit 1 and are set off by the bracket designated "A" on page 1.

On page 13 of Exhibit 4, and set off by the bracket designated "B", appears a discussion of imaging support provided by ADS. Specifically, this excerpt of text describes capturing and deploying images using "sysprep", which is part of the Windows Server OPK. On the third page of Exhibit 1, and appearing bracketed by a bracket designated "B" appears a discussion of how to create a master image. Notice in the discussion that the master installation is prepared with the "Sysprep" tool.

On page 13 of Exhibit 4, and set off by the bracket designated "C" is a discussion entitled "Task Sequences". This discussion indicates that a task sequence is a sequence of operations to be performed in order. The sequence definition is stored in an XML file on the controller. On page 8 of Exhibit 1 under the heading "Create task sequence file for image

1 employment" and bracketed by a bracket designated "C", appears a
2 discussion of how to create a task sequence file for the ADS controller.
3 This discussion describes the notion of creating a task sequence which is
4 an XML file containing a sequence of tasks for a controller to perform.

5
6 **The "Task Sequences Functional Specification" Document**

7 Turning attention to the "Task Sequences Functional Specification"
8 document (Exhibit 3), such indicates on its cover page that three of the
9 inventors were developer authors (indicated in the bracket).

10 On page 3 under the table of contents, a section designated 2.1.2.2
11 is entitled "Task sequences and Workflow Overview". In addition, a
12 section on page 4 designated 2.1.7 is entitled "Task sequence
13 Implementation". These sections correspond in content to content that
14 appears in Exhibit 1 beginning on page 8 under the heading "Create a job
15 template for the sequence file".

16 On page 6 of Exhibit 3 within the bracket designated "C" appears a
17 discussion pertaining to a task sequence. Such corresponds in content to
18 the discussion on page 8 of Exhibit 1.

19 On page 21 of Exhibit 3 appears a discussion the XML schema
20 including, at item "D", the task element which includes child elements
21 including the command and parameters element (item "E"). This
22 corresponds in content with the discussion on page 9 of Exhibit 1 which
23 illustrates a sample XML excerpt that includes the task and parameters
24 elements designated respectively, at "D" and "E". A discussion of the
25 parameters element occurs on page 22 of Exhibit 3.

1 On page 11 of Exhibit 3 appears a discussion, at item "F" of a job
2 template that can be created that refers to the task sequence. This
3 corresponds in content with the discussion on page 9 of Exhibit 1 at item
4 "F" entitled "Create a job template for the sequence file".

5 Based upon the similarities between Exhibits 3 and 4, which were
6 authored by subsets of the inventors, and Exhibit 1 - the ADS reference
7 cited by the Patent Office, as well as other similarities which are not
8 specifically identified above, it should be apparent that Exhibit 1 was
9 derived from and describes the work of the inventors as set forth in
10 Exhibits 3 and 4.

11 We further declare that all statements made herein of our own
12 knowledge are true and that all statements made on information and belief
13 are believed to be true; and further that these statements were made with
14 the knowledge that willful false statements and the like so made are
15 punishable by fine or imprisonment, or both, under 18 U.S.C. 1001, and
16 that such willful false statements may jeopardize the validity of the
17 application or any patent issued thereon.

18 
19 Curt A. Steeb

20 Co-Inventor of Application Serial No. 10/607,054
21
22
23
24
25

Munisamy Prabu
Co-Inventor of Application Serial No. 10/607,054

Zeyong Xu
Co-Inventor of Application Serial No. 10/607,054

Martin Holladay
Co-Inventor of Application Serial No. 10/607,054

Paul Sutton
Co-Inventor of Application Serial No. 10/607,054

Raymond Pedrizetti
Co-Inventor of Application Serial No. 10/607,054



Michael Gallop
Co-Inventor of Application Serial No. 10/607,054

Exhibit 1

Windows Server TechCenter > Windows Server 2003 Technical Library > Windows Server 2003: Deployment > Windows Server 2003: Deploy Server Clusters: Remote Setup, Unattended Installations and Image-based Installations

Image-based Installation of the Operating System and the Cluster Service Using Deployment Services (ADS)

Updated: January 01, 2003

Automated Deployment Services (ADS) enables you to personalize the operating system image to your needs. Imaging/Cloning requires sysprep.inf (file that contains the configuration information for operating system) every time you need to deploy the image to a different configuration of the operating system. ADS reduces this complexity by allowing the user to personalize the sysprep.inf file. About sysprep.inf and imaging please refer to section Image based installation of Server Cluster.

In addition, ADS enables you to use a single server to manage servers in your data center. This server, or "Controller," together with ADS services, enables you to deploy operating system images onto devices without an operating system or to repurpose existing deployment system images. In short, you can use ADS to:

- ① Remotely repurpose a device that has no operating system to a useful state or repurpose a device from one state to another state.
- Operate a scale-out data center by providing ability to run extensible and configurable operations, such as scripts, on one or more servers.

Automated Deployment Services consists of the following three main services:

- Controller service
- Image Distribution service
- Network Boot Services

Refer to ADS help and documentation for more information about ADS and ADS setup and installation. This section assumes that you

Following are the steps to deploying Server Clusters using ADS

1. Add devices (systems that you want to install cluster service to) to ADS Controller
2. Create a Master Image
3. Upload the image to ADS controller
4. Modify sysprep.inf to include Variables
5. Create task sequence file for Image deployment
6. Create a job template for the sequence file
7. Create and store variables associated value in ADS database for the desired device
8. Execute Job Template against the desired device

⚠ Caution:

If you are using shared storage device then it is vital that only one node have access to the shared disk. Otherwise the shared disk shutdown all but one node or use some other technique (LUN, zoning etc) to isolate the datapath for the nodes. Once a single node connects to this cluster. Here are the key things you need be aware and plan accordingly before deploying clusters.

- The disks must be preformatted and ready to mount when the machine boots up in to operating system.
 - No other machine should be in full operating system at the time when the first node is creating the cluster (this is to avoid any s
 - Join Node (adding another node to the cluster) task sequence must be run only after a cluster has been formed . This is to avoid i
- is still in progress.

1. Add Devices (Systems that You Want to Install Cluster Service to) to ADS Controller:

After installation and configuration of ADS, you would need to add device(system where you want to deploy operating system and the device through UI or through command line interface. Refer to *ADS Help* section *Manage Devices* for more information. You need BIOS. Once the device boots up, it communicates with Controller and boots into deployment agent. This device will show-up in the device. You can control the device through UI (ads.msc) or through command-line interface.

To add a device through command line interface:

```
adsdevice [/s controllername [/u username /w password]] /add devicename [/description "text"] [/ip ipaddress] [/adminnm  
[/assettag assettag] [/jobtemplate templatename] [/?]
```

or

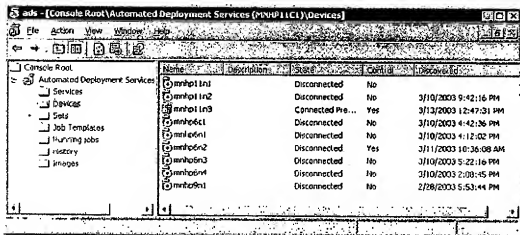
```
adsdevice [/s controllername [/u username /w password]] /add @filename
```

E.g.

```
Adsdevice /add mnhp6n11 /description "server cluster 1st node" /ip 172.24.11.207 /adminmac 00306E1215b8 /guid 239d1df  
00306e12 /jobtemplate servercluster.xml
```

To add a device through ADS.msc (UI interface):

Open Automated Deployment Services.



In the console tree, right-click the **Devices** folder and click **Add**.

In the **Add Device** dialog box, type the device name in the **Name** box.

If you want to add more information to help identify the device, type the information in the remaining boxes, and then click **OK**.

2. Create a Master Image:

Creating Master image consists of following steps:

- Building a master installation on a master computer. Building a master installation includes installing and configuring the operating include on your disk image.
- Preparing the master installation with the Sysprep tool. This includes configuring and running the Sysprep tool on the master comp
- Generating a disk image of the master installation with the disk-imaging tool. This includes saving each disk image to a permanent

☒ Note:

You cannot clone a cluster node with cluster service installed. You must de-install the cluster service or use a specially prepared m create a disk image.

Refer to Image based installation of Server Cluster for more information about creating master image.

3. Upload the Image to ADS Controller

After Capturing the image of the operating system you would need to add the captured image to the controller. Create directories n root of the controller. Make sure the volume where you create these directories have enough free disk space to hole at least 2 times

Next step is to add the image to the ADS controller database. You can add the image either through User Interface or command line image

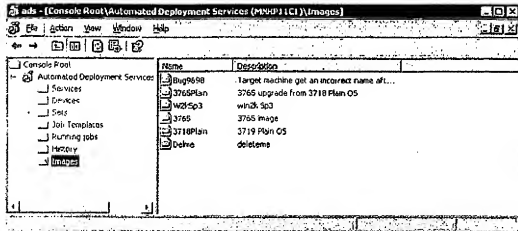
adsmage [/s controllername [/u username /w password]] /add imagename /path imagepath [/description]

After adding the image, you would need to move the image to the Image directory. You can check if the image is been properly add

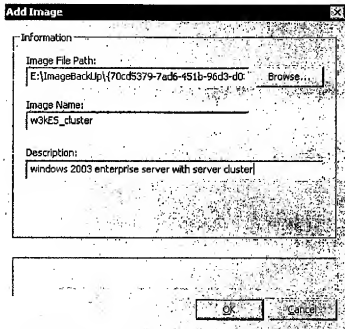
Refer to managing Images section in ADS help for more information about managing image.

To add the image through UI Interface

open ADS.MSC (Automated deployment service)



Right click **Images** and click **Add**



Fill out the Image File Path and Image Name, click **OK**

4. Modify sysprep.inf to include Variables

Sysprep.inf file is used by the mini-setup to install and configure the operating system. This file contains the configuration information for the operating system. This configuration information is static to the image. ADS allows you to customize sysprep.inf file by allowing you to remove the configuration information that will change from system to system can be removed from the sysprep.inf file. Instead variable can be used.

Before modifying the sysprep.inf file from the image, you would need to mount the image to a drive. ADS has tools that will allow you to mount the image using below command line syntax

```
imgmount /mount /w {Imagefilename} /d: driveletter
```


Change drive to the above drive letter and change directory to sysprep directory. There 2 ways to create or modify the sysprep.inf. Refer to Imaged-Based server cluster Installation section for more information about sysprep.inf.

Open the sysprep.inf file in notepad.exe. Locate variables for your installation that changes from a system to system. Define them a sysprep.inf file. For example, In the following snippet of sysprep.inf file, items marked with * at the beginning of the line can be def

```
[params.MS_TCPIP.Adapter01]
* DHCP="No"
* IPAddress="10.11.26.11,,172.24.11.141"
  SpecificTo=Adapter01
* SubnetMask="255.255.0.0,255.255.255.0"
* WINS="No"
;Adapter02 is used for public network
[params.MS_TCPIP.Adapter02]
  SpecificTo=Adapter02
* DefaultGateway="172.24.11.1"
* DHCP="No"
* IPAddress="172.24.11.205"
* SubnetMask="255.255.255.0"
* DNSServerSearchOrder="172.24.10.2,172.24.0.2"
* WINS="Yes"
* WINSserverList="157.55.254.201,157.55.254.203"
[GuiRunOnce]
;see section 2.2.1 in this document for a sample text of AssignDriveLetters.bat file
;see configuring cluster section and appendix B for createfs.vbs
*Command0=%systemdrive%\scripts\AssignDriveLetters.bat
*Command1 = "%windir%\system32\cluster.exe /cluster:SV-CLUSTER /CREATE /NODE:SV-NODE1 /USER:domain\user /PA
*Command2 = "%systemdrive%\ClusterInstallFiles\createfs.vbs SV-CLUSTER ClusterGroup SVFileShareResource E:\
```

After replacing these with variables, the snippets will look like the following. Notice that each variable name starts with ^ and ends ADS will not install and configure the operating system properly. Note: In the guirunonce section, above sample snippet replaces th can run your own scripts in this section also. After successful installation, Windows will run commands in guirunonce section on the services specified.

```
[params.MS_TCPIP.Adapter01]
  DHCP="^ADHCP1A"
  IPAddress="^AIPAddress1A"
  SpecificTo=Adapter01
  SubnetMask="^ASubnetMask1A"
  WINS="^AWINS1A"
;Adapter02 is used for public network
[params.MS_TCPIP.Adapter02]
  DefaultGateway="^ADefaultGateway2A"
  SpecificTo=Adapter02
  DHCP="^ADHCP2A"
  IPAddress="^AIPAddress2A"
  SubnetMask="^ASubnetMask2A"
  DNSServerSearchOrder="^ADNSServerSearchOrder2A"
  WINS="^AWINS2A"
  WINSserverList="^AWINSserverList2A"
[GuiRunOnce]
;Mount all volumes before form/join the cluster
Command0="^ACLUSTER_COMMAND_MOUNTA"
Command1=AssignDriveLetters^A
Command2="^ACLUSTER_COMMANDA"
Command3=FileShare^A
```

Once you have modified the sysprep.in file, unmount image using the below command line interface.

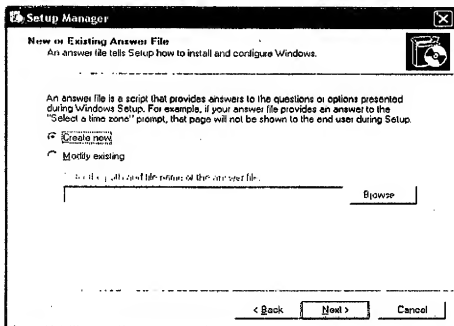
Imgmount u drive:

ADS also allows you to define the variables through scripts. Refer to ADS help on managing Images for more information.

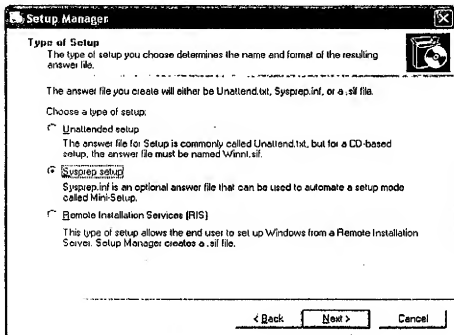
☒ **Note:**

Appendix F contains a modified and complete sample sysprep.inf.

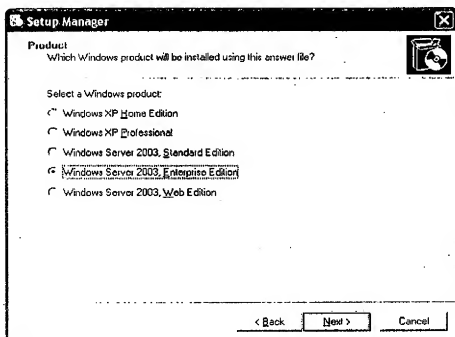
You can also use **setupmgr.exe** to create, modify unattended setup file, syspre.inf file with variables. Run setupmgr.exe through c



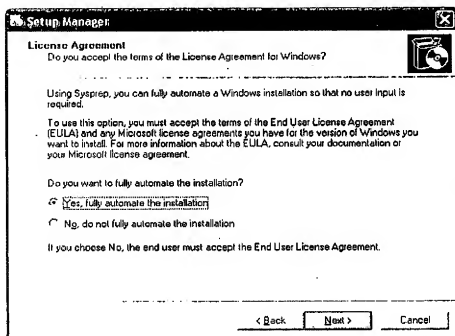
choose create new

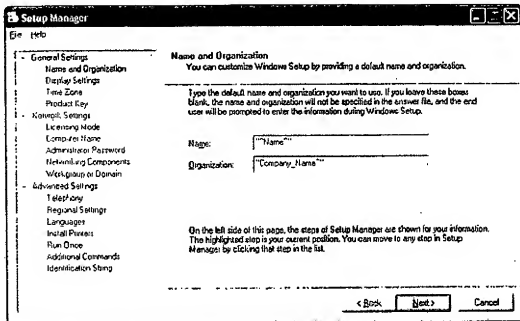


choose sysprep setup

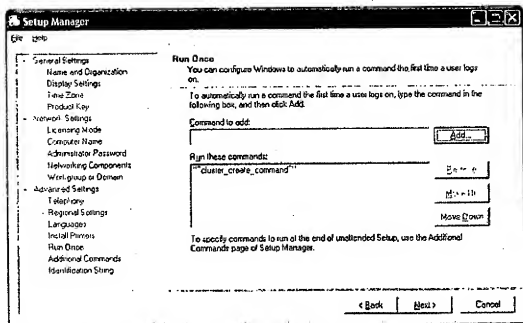


choose the operating system type that you want to deploy





Above allows you to enter values for the setup. Here instead of providing the actual values, you can provide the variable names that variable whose value will be replaced through ADS.




After completing above, click Run Once and enter the cluster configuration command. In the above example, ^cluster_create_command actually value will be defined in ADS. Once the sysprep.inf is created with the variables, save it and copy this sysprep.inf file into the

5. Create task sequence file for image deployment

Next step is to create a task sequence file for ADS controller. It is a XML file containing sequence of task for controller to perform a sample sequence files. You can use any of the sample XML file and modify to your needs. In this file you will define the personalize. In addition, you will define what other sequence of task controller must run against the device. For example, partition the disk, copy

reboot and check the state of the device etc.

You would need to modify at least the following sections in the sample XML files for it to work properly.



```

<!-- STEP 1 Create a single 4999MB partition on the disk -->
<task description="Partition the disk">
  <command>bmonitor/bmpart.exe</command>
  <parameters>
    <parameter>\device\harddisk0</parameter> <!-- selects harddisk0 -->
    <parameter>/init</parameter> <!-- erases all partitions on harddisk0 -->
    <parameter>/C:4999</parameter> <!-- creates a new partition (#1) of size 4999MB -->
    <parameter>/A</parameter>
    <!-- activate the newly created partition (#1) -->
  </parameters>
</task>

```

In the above section (step 1) you need to define the partition size for the disk. In the above example it is

```

<!-- STEP 2 download images -->
<task description="Download image">
  <command>/imaging/imgbndeploy.exe</command>
  <parameters>
    <parameter>3718Plain</parameter> <!-- name of the image to be deployed-->
    <parameter>\device\harddisk0\partition1</parameter> <!-- deploy the image to partition1 -->
    <parameter>-r</parameter> <!-- specifies deploy mode -->
    <parameter>-client</parameter> <!-- required parameter -->
  </parameters>
</task>

```

In the above section (step 2) you need specify the image name that you used to add to controller. In the above

```

<!-- STEP 3 Personalize the sysprep.inf file -->
<task description="Set sysprep custom info in the sysprep.inf file">
  <command>bmonitor/bmstprep.exe</command>
  <parameters>
    <parameter>\device\harddisk0\partition1\sysprep\sysprep.inf</parameter>
    <parameter>AProductKey</parameter> <!-- key(ProductKey) to be searched in sysprep.inf file -->
    <parameter>"$ProductKey$"</parameter> <!-- value to be replaced with. Make sure to Embed it in quotes -->
    <parameter>A OEMOuplicatiorstring</parameter>
    <parameter>"$OEMOuplicatiorstring$"</parameter>
    <parameter>A $CLUSTER_COMMAND_MOUNTS</parameter>
    <parameter>A $CLUSTER_COMMANDA</parameter>
    <parameter>A $CLUSTER_COMMANDS</parameter>
    <parameter>A AssignDriveLettersA</parameter>
    <parameter>A $AssignDriveLetters$</parameter>
    <parameter>A FileShareA</parameter>
    <parameter>A $FileShare$</parameter>
  </parameters>
</task>

```

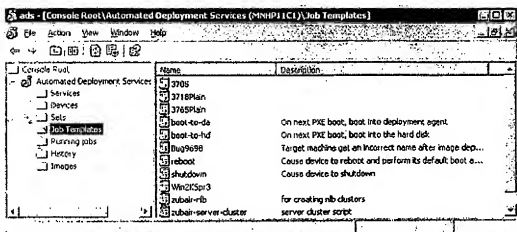
In the above section (step 3) you need to specify the variable names that you used in sysprep.inf. Notice that the variables are embedded replaced with is in quotes (\$xxxx\$).

A sample XML file for deploying a single node clusters is included in the appendix D. This sample file will also create file shares on the

5. Create a job template for the sequence file

Job templates in ADS provides a way to define and store instructions for task that you plan to run against a device (or set of device). **adsjobtemplate** command-line tool or the ADS snap-in to create a job template. You would need to associate this job template with operating system and configuring server cluster. After you create a job template, you can use it to run a job as often as you want a values. For instance, you can first create a single node cluster using the cluster command with create switch. After the cluster created system you want to join this cluster after operating system deployment. For this you would only need to change the value associated

To add the jobtemplate using ADS UI Interface. Open ADS.MSC

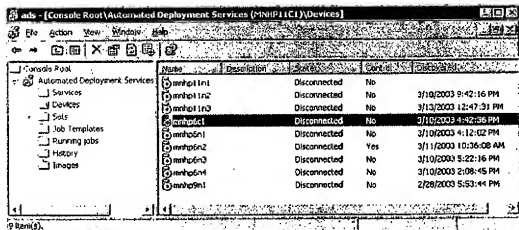


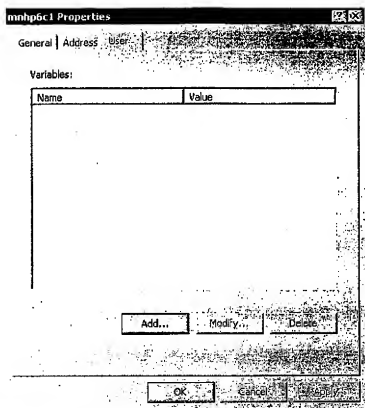
Right click Job Template and click Add. This will invoke the adding job template wizard that will guide you through the process.

7. Create and store variables associated value in ADS database for the desired device

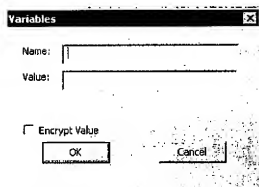
Now you would need to define and store the variables for each device in controller database. These values get replaced in sysprep.inf can define variables through ADS snap-in or command-line interface. To add through ADS Snap-in, double-click the device and on t name that is used in the sequence file and sysprep.inf and associate it with a value.

Open ADS.MSC and double click the appropriate device





Click User and then click Add .



Enter the variable name and then enter the value for this variable and click ok.

You can also use command-line Interface as listed below to add variables to the device database on the controller.

`Adsdevice /edit device-name /setvar variable-name "value"`

E.g.;

`Adsdevice /edit mnhp11n3 /setvar DNSServerSearchOrder2 "172.24.10.2,172.24.0.2"`

Mnhp11n3 is a device name, DNSServerSearchOrder2 is a variable.

You can create a batch script to add all the variables for a device. Appendix G includes a sample batch script file that defines values




8. Execute Job Template Against the Desired Device

Next step is to execute the job templates against the device. To assign a job template to devices manually, use either the **ADS snap-in**. Once the device is connected to the controller, right click the device and click run job. This will invoke a wizard that will take you through the execution.

Once the execution starts, you can monitor the job progress by clicking on running job option on the left pane of ADS snap-in. It lists XML file. After cluster creation you may want to configure the cluster based on your need. You can use any script to configure cluster.

You can use the same sequence file, sysprep.inf, Image for installing and configuring second node with little modification. You would add the second node(device). For example, cluster command to add a node to this cluster, IP address for the node etc.

Was this information helpful?

 [Printer-Friendly Version](#)  [Email this page](#)  [Add to Favorites](#)

[Manage Your Profile](#)

© 2006 Microsoft Corporation. All rights reserved. [Terms of Use](#) | [Trademarks](#) | [Privacy Statement](#)

Exhibit 2

Disclosure Packet

MS#: 302701.3

Title: Methodology for Task Sequence Scheduling

Microsoft Team: Andrew Sanders; Joan Bileau; Dianne Bouton
Inventors (names and email aliases): MS Prabhu [REDACTED], Mike

Gallop [REDACTED], Ray Pedrizetti [REDACTED], Curt Steeb [REDACTED], Zeyong Xu
[REDACTED], Paul Sutton [REDACTED], Martin Holladay [REDACTED]

Component/Subcomponent: SPG - BIG

Summary: A method of automatically installing operating systems on machines. This is accomplished by the use of task sequences which are XML-based instruction sets that are developed in accordance with the task sequence schema.

Formatted: Font color: Auto

Underlying Facts:

Formatted: Underline

Formatted: Underline

☐ No ☐ Unknown

☐ No

☐ No

Formatted: Underline, Font color: Auto

Standards Related?

☐ Yes

☐ No

Formatted: Font color: Auto

Formatted: Normal

Prior/Related Art: 302697.1, Parallel Asynchronous System for Automated Deployment, 302698.1, Mechanism to Interface with Task Sequence Engine

Attachments: Preferred OC Atty: (if we have a preference, based on certain OC atty's prior experience with a particular technology area of ours)



predelclosure



Job Design Engine



Task Sequence Spec



ADS Functional Overview

Last Updated: 4/25/2001 9:11 AM

Formatted: Normal

Exhibit 3

Windows

Task Sequences Functional Specification

Please note: In order to view the guidelines and suggestions under each heading in the document you must choose to view hidden text (click Show/Hide on the standard toolbar).

Microsoft Confidential. © 2001 Microsoft Corporation. All rights reserved.
FOR DISCLOSURE UNDER NDA ONLY

These materials are confidential to and maintained as a trade secret by Microsoft Corporation. Information in these materials is restricted to Microsoft authorized recipients only. Any use, distribution or public discussion of, and any feedback to, these materials is subject to the terms of the attached license. By providing any feedback on these materials to Microsoft, you agree to the terms of that license. If the license agreement has been removed, review the terms at <http://www.microsoft.com/forshare/relationship01.asp> before using these materials.

Feature Information	
Feature Name	Task sequences
Area	
Related Features	
Requirements Spec	
Document Location	
Spec Status	New
Document Security	Public (All MS) () Private (Only Jim's group) ()

Contact Information	
PM Author	PSutton
Dev Author	Curtis, ZeyongXu, MPrabu
Test Contact	JeffForm, JaiLee
Design	
Usability	TonyLan
UA	
PSS	

Microsoft Corporation Technical Documentation License Agreement (Standard)

READ THIS! This is a legal agreement between Microsoft Corporation ("Microsoft") and the recipient of these materials, whether an individual or an entity ("You"). BY ACCESSING, USING OR PROVIDING FEEDBACK ON THE ATTACHED MATERIALS ("Materials"), YOU AGREE TO THESE TERMS.

1. These Materials are Microsoft confidential information under Your most recent non-disclosure agreement with Microsoft ("Your NDA"). However, You may use these Materials only as described in Paragraph 2 below.
2. You may review these Materials only (a) as a reference to assist You in planning and designing Your product, service or technology ("Product") to interface with a Microsoft Product as described in these Materials; and (b) to provide feedback on these Materials to Microsoft. All other rights are retained by Microsoft; this agreement does not give You rights under any Microsoft patents. You may not (i) duplicate any part of these Materials, (ii) remove this agreement or any notices from these Materials, or (iii) give any part of these Materials, or assign or otherwise provide Your rights under this agreement, to anyone else.
3. These Materials may contain preliminary information or inaccuracies, and may not correctly represent any associated Microsoft Product as commercially released. All Materials are provided entirely "AS IS." To the extent permitted by law, MICROSOFT MAKES NO WARRANTY OF ANY KIND, DISCLAIMS ALL EXPRESS, IMPLIED AND STATUTORY WARRANTIES, AND ASSUMES NO LIABILITY TO YOU FOR ANY DAMAGES OF ANY TYPE IN CONNECTION WITH THESE MATERIALS OR ANY INTELLECTUAL PROPERTY IN THEM.
4. If You are an entity and (a) merge into another entity or (b) a controlling ownership interest in You changes, Your right to use these Materials automatically terminates and You must destroy them.
5. You have no obligation to give Microsoft any suggestions, comments or other feedback ("Feedback") relating to these Materials. However, any Feedback you voluntarily provide may be used in Microsoft Products and related specifications or other documentation (collectively, "Microsoft Offerings") which in turn may be relied upon by other third parties to develop their own Products. Accordingly, if You do give Microsoft Feedback on any version of these Materials or the Microsoft Offerings to which they apply, You agree: (a) Microsoft may freely use, reproduce, license, distribute, and otherwise commercialize Your Feedback in any Microsoft Offering; (b) You also grant third parties, without charge, only those patent rights necessary to enable other Products to use or interface with any specific parts of a Microsoft Product that incorporate Your Feedback; and (c) You will not give Microsoft any Feedback (i) that You have reason to believe is subject to any patent, copyright or other intellectual property claim or right of any third party, or (ii) subject to license terms which seek to require any Microsoft Offering incorporating or derived from such Feedback, or other Microsoft intellectual property, to be licensed to or otherwise shared with any third party.
6. Microsoft has no obligation to maintain confidentiality of any Microsoft Offering, but otherwise the confidentiality of Your Feedback, including Your identity as the source of such Feedback, is governed by Your NDA.
7. This agreement is governed by the laws of the State of Washington. Any dispute involving it must be brought in the federal or state superior courts located in King County, Washington, and You waive any defenses allowing the dispute to be litigated elsewhere. If there is litigation, the losing party must pay the other party's reasonable attorneys' fees, costs and other expenses. If any part of this agreement is unenforceable, it will be considered modified to the extent necessary to make it enforceable, and the remainder shall continue in effect. This agreement is the entire agreement between You and Microsoft concerning these Materials; it may be changed only by a written document signed by both You and Microsoft.

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/loisnspg/specs/sgmt01.asp>).

2001-01-01 MICROSOFT DOCUMENT IN 302701-4 - DISCLOSURE PACKET.DOCXAR SEQUENCES FUNCTIONAL SPEC.DOCXAR LAST SAVED: 2/3/2003 1:04:00 PM 2/3/2003 12:32 PM 03/02/02 4:55 PM 04/SEP02 4

Table of Contents

1 Scenario Description	8
1.1 Feature Summary	8
1.2 Theme Relevance	8
1.3 Goals	8
1.3.1 Non Goals	9
1.4 Scope	9
1.4.1 What does this spec cover	9
1.4.2 What this spec doesn't cover	9
1.5 Related Documents	9
2 Feature Description	9
2.1 Behavioral Specification	9
2.1.1 Terminology	9
2.1.2 Scenarios	9
2.1.2.1 The Problem	10
2.1.2.2 Task sequences and Workflow Overview	10
2.1.2.3 Handling for errors that occur in individual operations	12
2.1.2.4 Example 1 – Create User Account	13
2.1.2.5 Example 2 – Build New Server (PXE Environment)	14
2.1.2.5.1 Completed Scenario	191918
2.1.3 Overview	20
2.1.3.1 Schema Capabilities	20
2.1.3.2 Invoking on a Single Target Server	20
2.1.3.2.1 Errors	20
2.1.3.3 Invoking on Multiple Target Servers	21
2.1.3.3.1 Synchronization	22
2.1.3.3.2 Errors	22
2.1.3.4 Device Variables	22
2.1.3.5 Control Flow	22
2.1.4 XML Schema	22
2.1.4.1 Description	22
2.1.4.1.1 sequence	232323
2.1.4.1.2 task	23
2.1.4.1.3 command	23
2.1.4.1.4 parameters	242423
2.1.4.1.5 parameter	24
2.1.4.2 Formal Definition	2524
2.1.4.3 Examples	252524
2.1.4.3.1 Bare metal to Full OS	252524
2.1.4.3.2 Simple document	252524
2.1.4.3.3 Compound document with dtd	25
2.1.4.3.4 compound document with xslt	25
2.1.5 Jobs Objects Hierarchy	25
2.1.5.1 Hierarchies	25
2.1.5.1.1 One Operation on Single Device	25
2.1.5.1.2 One Operation on Multiple Devices	262626
2.1.5.1.3 Sequence on Single Device	26
2.1.5.1.4 Sequence on Multiple Devices	27

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/forining/spec/agmt01.asp>).

200535.DOC DOCUMENT IN 302701.1 – DISCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/3/2003 1:04:00 PM 2/2/2003 12:32 PM 03/OCT/02 4:55 PM 04/SEP/02 4

2.1.5.2	Summary of Types of Jobs Objects	282827
2.1.5.2.1	Examples	292928
2.1.5.3	Jobs Status	32
2.1.5.3.1	Per-Operation (Leaf-Node) Job State Definitions	33
2.1.5.3.2	Parent State Definition	363636
2.1.6	PXE Boot Actions and Default Sequences	363837
2.1.7	Task sequence Implementation	393938
2.1.7.1	Examples	393938
2.1.7.2	Task Sequence Architectural Description	414440
2.1.7.3	Reading a Task Sequence	434342
2.1.7.4	Implementation Notes	444443
2.1.7.4.1	WMI Interfaces	444443
2.1.7.4.2	Loading Task Sequence	444443
2.1.7.4.3	Executing Task Sequence	444443
2.1.7.5	Running a Task Sequence	454644
2.1.7.6	Implementation Notes	454644
2.1.7.6.1	APIs	454644
2.1.7.6.2	Workflows	454644
2.2	UI Description	474746
2.2.1	XML Schemas	474746
2.2.1.1	Task sequence Schema	484847
2.2.1.2	Variable Schema	484847
2.3	API and Interfaces	484847
2.3.1	New APIs and Interfaces	484847
2.3.2	Changed APIs and Interfaces	484847
2.3.3	Removed APIs and Interfaces	484847
2.4	Scripting Interfaces	484847
3	Integration Issues	484847
3.1	Manageability	484847
3.2	Security Impact	484847
3.2.1	Security Context	484847
3.3	Setup Requirements	484847
3.3.1	Component Isolation	484847
3.4	Hardware/Environmental Requirements	494948
3.4.1	Network Bandwidth	494948
3.4.2	Boot time	494948
3.4.3	Battery life	494948
3.5	Source File Impact	494948
3.5.1	Depots Affected	494948
3.5.2	New Trees	494948
3.6	Directory Impact	494948
3.6.1	New Objects	494948
3.6.2	Object Changes	494948
3.6.3	Object Deletions	494948
3.6.4	Performance	494948
3.7	Registry Impact	494948
3.7.1	New Registry Keys	494948
3.7.2	Registry Key Changes	505049
3.7.3	Registry Key Deletions	505049
3.8	Win64 Issues	505049
3.9	Localization/Globalization Issues	505049
3.10	Accessibility Issues	505049
3.11	Compatibility	505049

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/specs/agmt01.asp>)

209955.DOCXDOCUMENT IN 3027914 - DISCLOSURE PACKET.DOCX TASK SEQUENCES FUNCTIONAL SPEC.DOCX LAST SAVED: 2/3/2003 1:04:00 PM 03/23/2003 12:32 PM 03/OCT/03 4:55 PM 04/SEP/04

3.11.1	Hardware Compatibility	505049
3.11.2	Application/Component Compatibility	505049
3.11.3	Heterogeneous Network Compatibility	505049
3.12	Backup	505049
3.13	Reliability Issues	505049
3.14	URL requirements	505049
3.15	Supportability	515150
3.15.1	Logging/Eventing	515150
3.15.2	Error Messages	515150
3.15.3	Diagnostic Tools	515150
3.15.4	Recovery from Corruption or Error Conditions	515150
3.16	User Assistance Issues	515150
3.17	Key feature interactions	515150
3.17.1	Terminal Services	515150
3.17.2	Clustering	515150
3.17.3	Network Infrastructure	515150
3.17.3.1	Remote Access	515150
3.17.3.2	Multiple Hops	515150
3.17.3.3	Network Media	515150
3.17.3.4	Network Diagnostics Impact	525251
3.17.4	Embedded Considerations	525251
3.17.4.1	Footprint impact (memory, disk)	525251
3.17.4.2	Run-time install dependencies	525251
3.17.4.3	Component definition (SLD) overview	525251
3.17.4.3.1	Resources	525251
3.17.4.3.2	Group membership/relationships	525251
3.17.5	Headless Considerations	525251
4	Document Sensitivity	525251
5	Unsupported Features	525251
5.1	Per-Workflow State Variables	525251
5.2	Updating State Variables From Agent	535352
5.3	Synchronization Between Sequences on Different Targets	535352
5.4	Sequence Errors	545453
6	Issues	555554

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/spec/agmt01.asp>).

200955.DOC DOCUMENT IN 3927014 - DISCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/23/2003 1:04:00 PM 12/3/2003 12:32 PM 03/0C702-655.PMD 456PW42

Functional Specification

Task Sequences

1 Scenario Description

1.1 FEATURE SUMMARY

A task sequence is a sequence of steps to be run on one or more target systems. Each step is either an operation or another sequence. Operations things like running a script or program, and are defined in the Remote Execution Specification.

The IT Industry needs computing environments that are highly manageable so as to reduce the total cost of ownership. Industry support and common standards are critical in building such environments given that management operations usually cross hardware and software boundaries. ADS provides a way to run a task on one or more target servers. By itself this addresses only a single task at on one or more managed servers. If an end user wants to execute a sequence of tasks on one or more managed target servers without any user intervention, it can be achieved through defining and executing a task sequence.

A task sequence is a group of tasks that are to be sequentially executed on one or more target servers. This group is defined in one or more XML documents. Sequence author can even integrate multiple sequence xml document modules into a single sequence document using standard XML features like XSLT, XPath, etc available in MSXML 4.0. Since Xinclude, XPointer, XBase and XLink concepts are not supported in MSXML 4.0, XSLT and DTD entity referencing techniques are exploited to support the extensibility through the use of sequence modules.

1.2 THEME RELEVANCE

Task sequences are required for two main things. First, to enable the remote, unattended deployment of bare metal servers from no OS through to a configured and running operating system and application. This also includes re-purposing a current system as though it was a bare-metal system.

Second, to enable unattended sequences of remote operations against running OSES such as performing a series of steps to check for and if necessary install hotfixes.

1.3 GOALS

- Enable sequencing of operations.
- Any step that can be executed through a sequence could be executed as an individual operation.

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/specs/agmt01.asp>).

200935.DOCDOCUMENT IN 302701.1 - DISCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/2/2003 1:04:00 PM 2/3/2003 12:32 PM 03/03/02 4:46 PM 04/SEP/02

1.3.1 Non Goals

- Synchronization between steps on different target servers (e.g. no support for stopping sequence on other targets if sequence fails on one target, or for performing a given step at the same or serially on different targets).
- Transactional capabilities (e.g. no support for either all steps succeed or all fail).
- Adding operation types that are specific to sequences.

1.4 SCOPE

1.4.1 What does this spec cover

1.4.2 What this spec doesn't cover

1.5 RELATED DOCUMENTS

- [Remote Execution Functional Spec.doc](#)

2 Feature Description

2.1 BEHAVIORAL SPECIFICATION

2.1.1 Terminology

Operation – a single process initiated from the controller on one or more servers

Sequence – an ordered list steps to be executed, where each step is either an operation or another sequence

Task Sequence – see Sequence

Step – a single element in a sequence, which can be either an operation or a sequence.

Job – either something that the user initiates (for example, a sequence or an operation), or an individual particular operation

Jobs object – an object in the object model which can represent one of an operation, a sequence, or a set of sequences on multiple target devices (implemented as a record in the Jobs table in the database and a class in WMI).

2.1.2 Scenarios

ADS provides a way to run a task on one or more target servers. The task can be a windows program, a script, or various types of special actions (such as reboot or shutdown). Task sequences add a way to sequence a series of tasks so that they can be performed one after another without requiring administrator intervention to start each one manually.

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/spec/sgm01.asp>).

200955.DOC DOCUMENT IN 3027014--DISCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/3/2003 1:04:00 PM 3/3/2003 12:32 PM 03/OCT/02 4:55 PM 04/SEP/02 4

2.1.2.1 The Problem

As an example of why task sequences are useful, consider the case of creating a user account. In this example, the following steps must be performed:

- 1) Create a user account on the target server
- 2) Assign a disk quota for this user on the target server
- 3) Create a mailbox for this user on the target server

Without task sequences, the administrator has two ways to implement this.

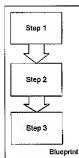
First, they could create a single script that performs all of these operations. This script can then be run from the controller, with the required input information (such as username and disk space quota to apply). However having a single script reduces the modularity of the system. If the administrator next has to create a provisioning process that comprises creating a user account, copying some files, and creating a mail box, they would have to create a new script. The administrator could re-use code from the first script, but now the administrator needs to modify both these scripts if a bug is found in the create a user part, for example. As the number of steps and number of different sequences increases, the system will get less reliable and ultimately be unmanageable.

A second way that the administrator could implement this without task sequences is by writing a separate script for each step. There would be a "create user" script, a "assign disk quota" script, and a "create mailbox" script. To actually provision the user, the administrator would go to the controller and run the first script. If that script succeeds, the administrator would run the second script. If that succeeds, the administrator would run the third script. This has the advantage over the first solution that the process is very modular. By running different scripts in various sequences the administrator can implement a lot of different provisioning processes. Each individual script is relatively small and self-contained, so it can be well tested and reliable. However there is additional workload on the administrator. They both need to manually run the scripts and check the results, and remember what scripts to run in what order for each provisioning process. Task sequences help the administrator by automating the sequencing of multiple steps.

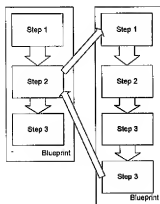
2.1.2.2 Task sequences and Workflow Overview

A task sequence is a definition of a sequence of steps. Each step can either be an operation supported by the controller (e.g. the ability to run a script on a target server), or another task sequence.

The following diagram shows a task sequence that defines a sequence of three steps, each of which is an operation:



This diagram shows a task sequence where the second step is a reference to a different task sequence:



A task sequence is defined in a file, using the task sequence definition schema. The task sequence definition file contains a section for each step in the task sequence. Each step may be a reference to another task sequence definition file, or a definition of an operation.

Operations are defined with the same attributes as defined in the Remote Execution Functional Spec.doc document. Specification, each operation is defined with:

- Delivery (how the content - if any - is obtained by the target)
- Command (for programs, this is the program to download or run, or for special actions, this is the action to perform)
- Parameters (any arguments to the program given in the command field)

For example, if the step is to download the script C:\CreateUser.vbs from the controller to the target device, and run it with the argument johndoe, the task sequence operation would be defined like this:

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/specs/agmt01.asp>).

200956.DOC\DOCUMENT IN 3027014--DISCLOSURE PACKET.DOC\TASK SEQUENCES FUNCTIONAL SPEC.DOC\LAST SAVED: 2/3/2003 1:04:00 PM 2/3/2003 12:32 PM 03XOC703.4.66 PM 04/SEP/04


```

Delivery      = BMCP
Command       = C:\CreateUser\bin
Parameters    = johndoe

```

This is given in a simplified syntax, the actual task sequence specification syntax is based on XML and is defined elsewhere in this document. It means that the operation is to run a script. The script is sent to the device using BMCP (the Delivery). The script path name, on the controller, is given (Command), and the parameters or arguments to the script (Parameters).

The actual syntax would be:

```

<task delivery="BMCP">
  <command>C:\CreateUser\bin\command</command>
  <parameters>
    <parameter>johndoe</parameter>
  </parameters>
</task>

```

However placing the name of the user account into the task sequence definition scheme like this is not very useful. Instead, it is possible to substitute variables into the individual operations. This can be done in two ways:

- First, variables can be substituted when the XML definition is read. This makes use of standard XML file parsing to replace formal parameters defined in the schema with values defined via a 'stylesheet'. This is called "XML substitution".
- Second, variables can be substituted when each job is started. These variables are substituted from information in the object model on the controller. In this version, the only variables available are those associated with a device record, and they can only be updated on the controller. However in the future, it will be possible to update the variables from the agent (so, for example, one stage of a sequence could use variable values set in a preceding stage), or use variables from different aspects of the object model, such as variables associated with images or with a currently running workflow. This is called "variable substitution".

The alternative to XML substitution is to use variable substitution. Here is an example step, in simplified syntax, showing the use of a variable. Variables can only be specified in the parameters field. When this sequence is loaded, the value of the Parameters field will be stored as given. When the step is to be run, the variable will be substituted by its value. In this example, the variable \$device.user.username will be replaced with the value of the user-defined variable with name "username" associated with the device on which this step is being run.

```

Delivery      = BMCP
Command       = C:\CreateUser\bin
Parameters    = $device.user.username

```

Task sequences can be considered as a higher level wrapper for one or more operations. Besides the definitions of the individual operations, task sequence feature adds:

■ 2.1.2.3 Handling for errors that occur in individual operations

The task sequence is defined without reference to any target servers. It is "invoked" against one or more target servers. It is assumed in this version that if it is run on multiple target servers, then all the steps run on all target servers. Each sequence is a serial list of steps,

Formatted: Heading 4, No bullets or numbering

executed in order. There is no control flow. If any step cannot be started or fails to complete, the sequence will stop (on that target server).

When a sequence is executed on multiple targets, it runs in parallel on each target. In this version, there is no communication between the sequences running on different targets, so there is no synchronization between targets.



A job template can be created that refers to the task sequence. This template may also refer to a device or set, as well as a task sequence parameters file.

2.4.2.32.1.2.4 Example 1 – Create User Account

In this example, three steps are needed:

- 1) Create a user account
- 2) Assign a quota
- 3) Create a mailbox

An operation is defined for each of these, comprising a script to be run on the target server. The following scripts are written:

- `CreateUser.vbs username`
- `AssignQuota.vbs username quota`
- `CreateMailbox username`

The text in *italics* shows the parameters required for each script.

These jobs can be run individually by the administrator, for example

```
C:\> SAJob -run -device server01.hoster.com -path c:\CreateUser.vbs johndoe
Job started with job id 76

C:\> SAJob -result 76
Device                               Status
-----
server01.hoster.com                  Running

Wait for job 76 to complete, error out, or timeout.

C:\> SAJob -run -device server01.hoster.com -path c:\AssignQuota.vbs johndoe 50
Job started with job id 76

C:\> SAJob -summary -jobid 76
Device                               Status
-----
server01.hoster.com                  Running

Wait for job 76 to complete, error out, or timeout.

C:\> SAJob -run -device server01.hoster.com -path c:\CreateMailbox.vbs johndoe
Job started with job id 80

C:\> SAJob -summary -jobid 80
Device                               Status
-----
server01.hoster.com                  Running
```

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/finances/sg/specslagm01.asp>).

200956.DOC\DOCUMENT\IN-3027014--DISCLOSURE PACKET\DOCTASK SEQUENCES\FUNCTIONAL SPEC\DOCLAST.SAVED: 2/3/2003 1:04:30 PM/2/3/2003 12:32 PM/03/02/2003 4:55 PM/04/SEP/02/


```

Server1: hoster.com          Running
-----
Wait for job 50 to complete, %ERRORLEVEL% is 0

```

Alternatively, a task sequence definition can be created. The specific syntax for the definition is XML, and is given later in this specification. Here it is given in a pseudo-language.

```

SPS1
  Delivery = SWP1000
  Command = C:\CreateUser.vbs
  Parameters = %device.user.username
Step 2
  Delivery = SWP1000
  Command = C:\SetUserProperties
  Parameters = username %device.user.username --quote %device.user.quote
Step 3
  Delivery = SWP1000
  Command = C:\CreateMailbox.vbs
  Parameters = %device.user.username

```

Note:

- For task sequence variables, the formal variable is replaced with the value when the task sequence is invoked. For device variables, the formal variable is replaced with the value when the specific operation is started on a particular target server.

2.4.2.42 Example 2 – Build New Server (PXE Environment)

Formatted: Bullets and Numbering

Task sequences are particularly useful for building a new server from bare metal. Reboots are required in the process, so this cannot be implemented as a single script.

This example assumes that the new server supports PXE and the environment supports PXE requests (that is, there is a DHCP server that responds with appropriate PXE tags).

A task sequence can now be started for this server. The server is only identified by its MAC address on the controller. The administrator would locate this record, and then start the task sequence.

The task sequence might do the following:

- Download and run a virtual floppy which configures the BIOS.
- Download and run a second virtual floppy, which configures the RAID array and some disk parameters
- Boot into the deployment agent and download an image to the primary hard drive – this will be the production OS
- Personalize the newly downloaded image with a hostname, domain name, IP address and possibly other information
- Boot into the production OS now on the hard drive
- Possibly perform further configuration steps in the production OS, such as installing applications or utilities.

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/spec/agrm01.asp>).

200556.DOC DOCUMENT IN 302701.1 – DISCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/2/2003 1:04:00 PM 2/3/2003 12:32 PM 03/OCT/02 4:55 PM 04/SEP/02 4

At a high level, this task sequence comprises three sections. The first section is instructions to be done during the PXE boot phases. At the end of this phase, the task sequence marks the device record to tell it the next boot it to be into the deployment agent. The second phase is the operations run against the deployment agent. At the end of this phase, the device record is updated to tell the controller to let the system boot into the production OS. The third phase contains the operations to perform once the production OS is running.

So the complete list of actions that will be defined in the task sequence are:

- PXE environment:
 - 1) Run virtual a floppy to configure the BIOS
 - 2) Specify to boot to deployment agent on next boot
- Deployment agent:
 - 3) Partition the disk
 - 4) Run an operation on the device to download the target image
 - 5) Configure the downloaded image so it can talk to the controller (for beta 1)
 - 6) Personalize sysprep.inf on the target
 - 7) Specify to boot to hard disk on next boot
 - 8) Reboot
- Target OS:
 - 9) Perform an unattended install

Each of the steps is defined like this:

- Boot to vfloppy1

```
Step 1:
  Command: 1 /pxefloppy.v1
  Parameters: configbios.img
```

Here the name of the vfloppy file is "configurebios.img", which exists on the TFTP server on the system running the deployment agent builder server.

This waits until a PXE request comes in from the device. The device record must already have been created, and the PXEBootAction property set to 'Respond'. This sequence will have been associated with the device as the sequence to run when a PXE request is seen from the device, using the 'JobTemplate' property.

This assumes that the device is booted by hand. If the device is already running and has the agent install, this step could be preceded by a "reboot" step.

Note that now, when the server boots, the controller will tell the NBS to download and run this specified virtual floppy image. The task sequence will move immediately onto the next stage.

The virtual floppy executes without any communication with the controller. At the end of running, the virtual floppy code must do a reboot. If it does not reboot, the target device will be uncontrollable from the controller. The controller has no direct way to determine if the virtual floppy executed successfully. However the user might design a floppy to write state to a network share, and use the next step of the sequence to run a controller program that checks the status, and if necessary aborts the sequence.

This step is in the running state until the next PXE request is received (that is, the request initiated after the virtual floppy completes and does a reboot).

- Boot to deployment agent

When the virtual floppy reboots, the next step tells the PXE server how to respond to the PXE request. It tells it to respond with information necessary to run the deployment agent pre-OS in memory.

```
Step 2:  
Command      //X2/boot-da
```

This step now waits until the target system is available and running in deployment agent. It is an error if the system boots into a full OS. Once the system is available (which the controller notices either through auto-discovery or by polling), move onto the next step.

Deployment agent environment:

When the system boots into the deployment agent, it will announce itself to the controller. The controller will automatically take control of it because there is a task sequence job running for this device. By taking control, the controller establishes a secure link to the deployment agent that can be used to run additional steps of the sequence.

- Partition the disk

The target device is now running the deployment agent and is ready to accept commands from the controller. The first step is to partition the disk. In this sequence, a single partition of 3GB. Note that the image must have been captured from a volume of less than 3GB.

```
Step 21:  
Command      = //BMD10R/bmpart.exe  
Parameters   = /device\\harddisk0\partition0  
             /int:  
             /C:3000  
             /A
```

- Run an operation on the device to download the target image

The next step is the image download.

```
Step 4:  
Command      = //TDCING/ntofndeploy
```

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/specs/agmt01.asp>).

200955 DOCX DOCUMENT IN 3027014 - DISCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/3/2003 1:04:00 PM 2/3/2003 12:32 PM 4/3/02 4:55 PM 4/SEP/04


```
Parameters: w2k-sp2-base
/Device/\\.\harddisk\partitions
%*
/1190
```

This step is running until the job returns an exit status (or error). This will cause the image with name 'w2k-sp2-base-image' in the controller Images database to be downloaded from the image store to the target device.

- Configure the downloaded image so it can talk to the controller

The image is now on the disk, but is not configured to be able to talk to the controller. It needs to know at least the controller's IP address and port, and the public certificate of the controller. Possibly other information will be required as well, such as the NIC to use to communicate with the controller.

For beta 1, the image must include this information. In particular, it must include the controller certificate, which can be supplied when installing the ADS agent component.

Formatted: Bullets and Numbering

After beta 1, this will be set via a "personalize" step in the task sequence.

Post-beta-1 image configuration might be set in a different way, possibly implicitly.

In beta 1, this involves two explicit steps. The first supplies the controller certificate to the image. The certificate itself must be converted to a base64 encoded version and supplied as an argument.

```
Step 1:
Command: /PHOTOGRAPHY\imagestore\
Target: %*
/1190
```

The second step is to set the controller IP address and port.

Need syntax for setting controller IP address and port.

```
Step 2:
Command: /PHOTOGRAPHY\imagestore\
Target: %*
```

Formatted: Strikethrough

- Personalize the sysprep.inf file

The information applies to beta 1. After beta 1, this configuration will be done by downloading a sysprep.inf file from the controller as part of the "personalize" step in the task sequence.

Formatted: Bullets and Numbering

The image is now on the disk, but contains only the generic sysprep.inf. The sysprep.inf on the target device must now be personalized. This assumes that the sysprep.inf file in the image is properly prepared, with the values set to placeholders

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/loisening/specs/agmt01.asp>).

200955.DOC DOCUMENT IN 302701.4 - DISCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/3/2001 1:04:00 PM 2/3/2001 12:32 PM 03/OCT/02 4:55 PM 04/SBP/024

such as "BIG_ADMIN_PASSWORD" which will be replaced with actual values in this step (in this example, the hostname will be set to 'server5.hoster.com').

```
Step 7:
Command      = /X/INIT/mini-setupprep.exe
Parameters   = /G:0001(harddisk0:partition1)/sysprep/sysprep.inf
              "BIG_ADMIN_PASSWORD"
              password
              "BIG_PASSWORD_PASSWORD"
              "0000-0000-0000-0000-0000"
              "BIG_COMPUTER_NAME"
              server5.hoster.com
              "BIG_JOIN_WORKGROUP"
              hoster.com
              "BIG_DRIVE_LETTERS"
```

Note that the password is given in plain text in the task sequence here. The sequence file should be protected by ACLs to ensure that only authorized people have access to its contents.

- Reboot

```
Step 8:
Command      = /X/INIT/reboot
Job will reboot = True
```

Tell the device to reboot, then move to next step. This starts the mini-setup process on the device.

- Boot from hard disk

```
Step 9:
Command      = /X/BOOT/hd
Job will reboot = True
```

This waits until PXE request comes in, then send back information to tell device to boot to hard disk. It then continues to wait until discovery comes in, then moves to next step. This will start mini-setup running on the target device (assuming that the image deployed was captured after running sysprep). At the end of mini-setup, it will reboot again, so a second boot action is required.

- Boot from hard disk

```
Step 10:
Command      = /X/BOOT/hd
```

This causes the boot at the end of mini-setup to be directed to boot from the hard disk.

Target OS:

- Run an unattended install

```
Step 11:
Command      = c:\windows\setup\setup.exe
Parameters   = /X/UNCL:"feature1,feature2"
```

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/learning/spec/agmt01.asp>).

20055.DOC DOCUMENT IN 392791.1 - DISCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/2/2001 1:04:00 PM 02/02/2001 12:30 PM 1002 466 PM 04/SEP/01

The complete scenario for doing a bare-metal to full OS boot for a new server is given in this section.

First, the relevant images need to be created and made available.

1) Create an image of the production OS to be deployed.

This must be a sysprep'ed OS volume. This is stored on the image server through the controller. The details of capturing an image and placing it onto the image server are described in the imaging deployment specification. The sysprep.inf file must be prepared.

2) Create an image of the virtual floppy

Create a bootable physical floppy that runs the tools necessary to configure the BIOS. This floppy must run unattended, and take no input. Capture the physical floppy to a file, and place the file in the TFTP directory on every NBS server.

Details of creating the floppy image are given in the virtual floppy specification.

Second, the sequence needs to be created and made available for use. The task sequence looks like this:

```
<xml version="1.0" encoding="UTF-8" ?>
<sequence command="formatdisk" unit="disk" url="schemas.microsoft.com/sequence"
  task="download" time="1" timeout="10" description="Run virtual floppy"
  <command>pxe/boot/vx/configurebios.ing/commands
  </command>
  </task>
  <task description="Boot to deployment agent"
    <command>pxe/boot-ds/commands
    </command>
    </task>
    <task timeout="600" description="Partition the disk"
      <command>DISKPART /script: %command%
      </command>
      <parameters>
        <parameter>device=harddisk0 partition=0</parameter>
        <parameter>/unit=parameters</parameter>
        <parameter>/G=3000</parameter>
        <parameter>/s</parameter>
      </parameters>
    </task>
    <task timeout="999" description="Download image"
      <command>WGET /image/iso/efi/commands
      </command>
      <parameters>
        <parameter>url=sp2-base/parameters</parameter>
        <parameter>device=harddisk0 partition=0</parameter>
        <parameter>/unit=parameters</parameter>
        <parameter>/clients=parameters</parameter>
      </parameters>
    </task>
    <task timeout="600" description="Setting sysprep custom info in the
      sysprep.inf"
      <command>DISKPART /script: %command%
      </command>
      <parameters>
        <parameter>device=harddisk0 partition=0 sysprep/sysprep.inf</parameter>
        <parameter>url=ALM100ASW000</parameter>
        <parameter>/password=parameters</parameter>
        <parameter>/sysprep=windows@999999999999</parameter>
        <parameter>/XXXX-XXXX-XXXX-XXXX-XXXX</parameter>
        <parameter>/tag=OCTOBER-2000</parameter>
      </parameters>
    </task>
```

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/learning/spec/agrm01.asp>).


```

<parameter>server hoster.com</parameter>
<parameter>file join WORKGROUP </parameter>
<parameter>hoster.com</parameter>
<parameter>file join LOCAL </parameter>
</parameters>
</task>
<task doesReboot="true" description="reboot">
<comment>PXE/BIOS/reboot</comment>
</task>
<task doesReboot="true" description="boot to hard disk - starts mini-setup">
<command>pxe/boot/bd/command</command>
</task>
<task description="boot to hard disk - starts full OS">
<command>pxe/boot/hdc/command</command>
</task>
<task description="install application"></task>
<command>c:\uninstall\package.msi</command>
</parameters>
<parameter>ADDLOCAL="features,features2"</parameter>
</parameters>
</task>

```

2.1.3 Overview

2.1.3.1 Schema Capabilities

2.1.3.2 Invoking on a Single Target Server

When the first step has finished, the controller starts the second one. And so on for the third step. A step is regarded as having completed when the controller status for that job is updated from "in progress" to "completed" or "completed with errors".



2.1.3.2.1 Errors

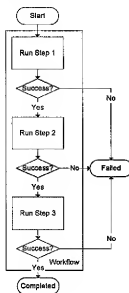
If an error occurs in a particular step, the sequence stops. An error can be one of these:

- The script cannot be run on the target server for any reason (e.g. network problems, target server not responding, controller out of resources, target server cannot run a script in this language).

- The script runs but does not complete (it is killed by the administrator from the controller, it is killed by the administrator on the target server, it is dies on the target server because of lack of resources, the target server shuts down or reboots before the script completed, etc).
- The script does not complete within a certain period of time (the timeout). The timeout is configurable for each operation. It should also be possible to configure the system to attempt to kill the process which timed-out (although this may fail).
- The script completes but returns an exit status that is defined by the task sequence system to mean "abort entire workflow". This occurs if the exit status is any value apart from zero.

If the controller aborts a workflow on a particular target device, it must log this information and maintain appropriate records in the object model to allow the administrator to determine what stage failed and which the error was. The administration can then either decide to fix the program and run the sequence again, or restart the sequence from the failed step

The diagram below shows the workflow. For the sake of simplicity, all the possible errors are expressed by the single "Success?" decision box. In reality, there are at least three places where a single step can fail: when it is being started, while it is running and after it has completed.

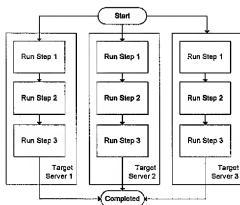


2.1.3.3 Invoking on Multiple Target Servers

A task sequence can be invoked on multiple target servers. The diagram below shows a task sequence that consists of three steps being invoked on three target servers. Error condition processing has been removed from the diagram.

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/learning/spec/sgm01.asp>).

200966.DOC\DOCUMENT-IN-3927914-DISCLOSURE PACKET.DOC\TASK SEQUENCES FUNCTIONAL SPEC.DOC\LAST SAVED: 2/3/2003 1:04:00 PM\2/3/2003 12:32 PM\OCT02 4:55 PM\04/SEP/02 4



2.1.3.3.1 Synchronization

In this version, there is no synchronization between the state of the sequence on different target servers. (See section 5.3).

2.1.3.3.2 Errors

In the single target server case, if an error occurs in a workflow, the workflow is aborted. When a task sequence is running on multiple target servers, the action to take on an error on a single target server could be:

- Abort the workflow on this target server only (others will continue to run)

Other actions have been punted. (See section 5.4).

2.1.3.4 Device Variables

Variables are defined in the Remote Execution Specification.

2.1.3.5 Control Flow

Not supported.

2.1.4 XML Schema

2.1.4.1 Description

This section describes each element that can be part of a task sequence.

The general format of a task sequence is:

```

<?xml version="1.0" encoding="utf-8"?>
<sequence xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.microsoft.com/windows/tasks" >
  <task sequence="..." />
</sequence>
  
```

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/spec/agmt1.asp>).

200555.DOC DOCUMENT IN 3027014 - DISCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/3/2003 1:04:00 PM 2/3/2003 12:35 PM 03/OCT/02 4:55 PM 04/SER/02 4

<sequence> element

The sections below describe each of the elements used within sequence definitions.

2.1.4.1.1 sequence

Formatted: Bullets and Numbering

Parent elements: None (if this sequence element is the root level element) or sequence

Child elements: task, sequence

Attributes:

- command – comment text (optional), by convention this typically contains the filename of the XML file
- parameters – comment text (optional)
- description – comment text (optional)
- xmlns – specify the default namespace for child elements, must have value "urn:schemas-microsoft-com:sequence"

Content: None

2.1.4.1.2 task

Formatted: Heading 5

Formatted: Bullets and Numbering

Parent elements: sequence

Child elements: command, parameters

Attributes:

- timeout – specify the timeout period for this step, in seconds (optional; default is 0 meaning no time out)
- doesReboot – 'true' means that this step will cause a reboot of the device. 'false' means it will not (optional; default is 'false')
- description – comment text (optional)

Formatted: Bullets and Numbering

Content: None

Formatted: Body Text

2.1.4.1.3 command

Parent elements: task

Child elements: None

Attributes: None

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/learning/specs/agm01.asp>).

209655.DOCDOCCLIMENT IN 3027914 - DISCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/3/2003 1:04:00 PM 2/3/2003 12:32 PM 03/02/02 4:46 PM 04/SEP/02 4

2.1.4.12.1.4.2 Formal Definition

Formatted: Bullets and Numbering

The following listing shows the task blueprint sequence XML schema defined with XSD (XML Schema Definition).

To be provided.

2.1.4.22.1.4.3 Examples

Formatted: Bullets and Numbering

2.1.4.2.12.1.4.3.1 Bare metal to Full OS

To be provided.

2.1.4.2.22.1.4.3.2 Simple document

Formatted: Bullets and Numbering

The following listing shows the simple blueprint xml document.

To be provided.

2.1.4.2.32.1.4.3.3 Compound document with did

Formatted: Bullets and Numbering

To be provided.

2.1.4.2.42.1.4.3.4 compound document with xsit

Formatted: Bullets and Numbering

To be completed.

2.1.5 Jobs Objects Hierarchy

Whenever a job is run, it will create one or more Jobs objects in the object model. These objects are used to represent the state of the job as it runs, and after it has run, the Jobs object represent the historical record of what the job did.

2.1.5.1 Hierarchies

This section describes the hierarchy of Jobs objects created and used for each type of job. There are four types of jobs that are of interest:

- Running a single operation against a single device
- Running a single operation against more than one device
- Running a sequence against a single device
- Running a sequence against more than one device

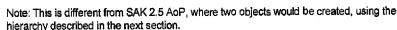
The Jobs objects used in each of these is described below.

2.1.5.1.1 One Operation on Single Device

A single Jobs object is created. This is called a "per-operation" Jobs object.

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/specs/agmt01.asp>).

200955.DOC\DOCUMENT IN 302701.1- DISCLOSURE PACKET\DOCTASK-SEQUENCES\FUNCTIONAL SPEC.DOC\LAST SAVED: 8/3/2003 1:54:00 PM\2/3/2003 4:12:32 PM\03/OCT/02 4:55 PM\04/SEP/02 4

[illegible]

This diagram shows this arrangement. The operation is run on a set containing m devices (where $m > 1$), named D_1 through D_m .

m Jobs objects are created, each one representing the status of the job on a particular device. These objects are called "per-operation" Jobs objects. They are direct children of the "multi-device single-operation" Jobs object.

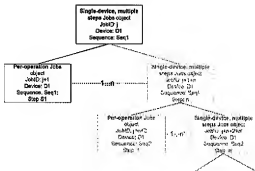


This diagram shows this arrangement. The sequence Seq1 comprises n steps (where $n \geq 1$), labeled S1 through Sn.

n Jobs objects are created, each one step of the sequence. These objects represent either a single operation on a single device (a "per-operation" Jobs type) or a sequence on a single device (a "single-device multiple step" Jobs object). These objects are direct children of the "single-device multiple step" Jobs object.

If any of these child Jobs represents a sequence, that sequence also has children jobs. In the diagram, a prototype step n of Seq1 is assumed to be another sequence. This other sequence is called Seq2 and has n' steps. Any of the steps in this sequence may itself be another sequence, so the hierarchy of Jobs objects may continue to indefinite depth.

200955.DOC DOCUMENT IN 302701.1 - DISCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/3/2003 1:04:00 PM 2/3/2003 12:32 PM 03/OCT/02 4:55 PM 04/SEP/02



2.1.5.1.4 Sequence on Multiple Devices

The structure is the same as above, but now there is a parent Jobs object to represent the state across all devices on which the sequence is being run.

This diagram shows this arrangement. The sequence is run on a set of m devices, named $D1 \dots Dm$. The sequence Seq1 comprises n steps (where $n \geq 1$), labeled S1 through Sn.

A Jobs object is created to represent the overall status sequence across all devices. This is called a "multi-device multiple step" Jobs object.

m Jobs objects are created to represent the overall status of the sequence on each individual device. These are called a "single-device multiple step" Jobs objects.

Under each of the m single-device, multiple step Jobs objects, n Jobs objects are created, each one representing a single step of the sequence. These objects represent either a single operation on a single device (a "per-operation" Jobs type) or a sequence on a single device (a "single-device multiple step" Jobs object). These objects are direct children of the "single-device multiple step" Jobs object.

If any of these child Jobs represents a sequence, that sequence also has children Jobs. In the diagram, a prototype step n of Seq1 is assumed to be another sequence. This other sequence is called Seq2 and has n' steps. Any of the steps in this sequence may itself be another sequence, so the hierarchy of Jobs objects may continue to indefinite depth.

Parent: None.

Direct Children: One "per-operation" Jobs object per device on which the operation is run.

- "Per-operation" Jobs object

This type represents the status of a single operation running on a single device. It has no children.

Parent: None, or "multiple-device, single operation" or "single-device, multiple steps".

Direct children: None.

Since the per-operation type has no children, it is also sometimes called a "leaf-node" in the Jobs tree. The other types always have children, and are called "parent nodes". The top-level node in any hierarchy (the one with no parents) is called the "root node".

2.1.5.2.1 Examples

The following diagrams show some example Jobs object hierarchies. The sequence definitions are given in a compressed pseudo-code.

In all these examples, the next available JobID is assumed to be 1, and it is assumed that each Jobs object created gets the next JobID in sequence. In practice, the next available JobID may be any value, and JobIDs may not be sequential, however they will always be in the same order.

2.1.5.2.1.1 Example: Single Operation on One Target

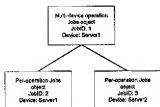
If a single operation (such as run a script) is run against a single server, Server1, it creates a Jobs hierarchy like this (assuming that the next available JobID is 1).



Since this is run on a single target, only a single Jobs object is required to hold the complete status of the job.

2.1.5.2.1.2 Example: Single Operation on Multiple Targets

If a single operation is run against the two devices called Server1 and Server2, it creates a Jobs hierarchy like this (assuming that the next available JobID is 1).

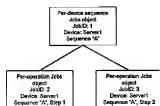


2.1.5.2.1.3 Example: Simple Sequence on One Target

Task sequence "A" consists of two steps:

- Step 1 – run a script
- Step 2 – run a script

If this sequence is run against the device Server1, it creates a Jobs hierarchy like this (assuming that the next available JobID is 1).



Since this is run on a single target, there is no "multi-device sequence" Jobs object to represent the status across multiple devices.

2.1.5.2.1.4 Example: Sequence Hierarchy on One Target

Task sequence "A" consists of two steps:

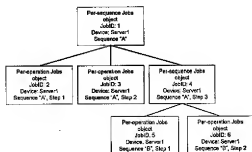
- Step 1 – run a script
- Step 2 – run a script
- Step 3 – run task sequence "B"

Task sequence B consists of two steps:

- Step 1 – run a script
- Step 2 – run a script

28

If sequence "A" is run against the device Server1, it creates a Jobs hierarchy like this (assuming that the next available JobID is 1).



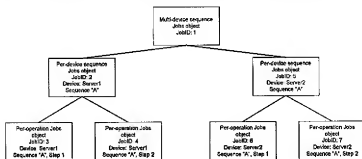
The third step is another task sequence, so rather than being a per-operation Jobs object, the third step is a per-sequence Jobs object, which represents the status of sequence "B".

2.1.5.2.1.5 Example: Simple Sequence on Multiple Targets

Task sequence "A" consists of two steps:

- Step 1 – run a script
- Step 2 – run a script

If this sequence is run against a set that contains two devices called Server1 and Server2, it creates a Jobs hierarchy like this (assuming that the next available JobID is 1).



2.1.5.2.1.6 Example: Sequence Hierarchy on Multiple Targets

Task sequence "A" consists of two steps:

- Step 1 – run a script

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/financing/specs/legm01.asp>).

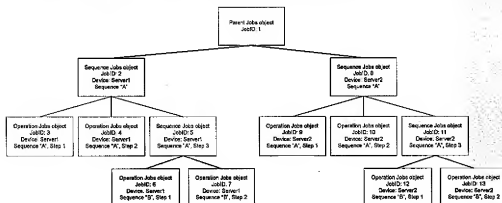
200955.DOCXDOCUMENT IN 3027014--DISCLOSURE PACKET.DOCX TASK SEQUENCES FUNCTIONAL SPEC.DOCX LAST SAVED: 20/2003 1:04:00 PM 03/2003 4:32:32 PM 03/2003 4:32:32 PM 04/2004 4:32:32 PM 04/2004 4:32:32 PM

- Step 2 – run a script
- Step 3 – run task sequence "B"

Task sequence B consists of two steps:

- Step 1 – run a script
- Step 2 – run a script

If sequence "A" is run against a set that contains two devices called Server1 and Server2, it creates a Jobs hierarchy like this (assuming that the next available JobID is 1).



2.1.5.3 Jobs Status

Add Titled Out status to jobs state? – RAUL/PABU/ZEYONG

Each Jobs object contains four fields used for status information. These are:

- State
The current state of the operations that this Jobs object represents.
- Children Count
The number of direct children of this Jobs object.
- Children Completed
The number of direct children of this Jobs object that have completed.

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/spec/agmt01.asp>).

200955.DOC DOCUMENT IN 302701.4 – DISCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/3/2003 1:04:00 PM 2/3/2003 12:32 PM 03/QCT/02 4:55 PM 04/SEP/04

20

■ Children Errors

The number of direct children of this Jobs object that are in an error condition (also called "errored").

The specific meaning of these four fields for each type of Jobs object is given below:

Type of Jobs Object	State	Children Count	Children Completed	Children Errors
Per-operation	Created Ready to Run Unable to Start Running Completed Stopped Failed Canceled Timed Out	Not used	Not used	Not used
Multi-device, single operation	Created Ready to Run Running Completed Failed/Errored	Total number of devices on which operation is to be run (count of direct children)	Number of devices which have completed successfully (count of direct children where status is set to Completed)	Number of devices on which the operation did not start or errored (count of direct children where status is set to Unable to Run, Stopped, Failed, Canceled or Timed Out)
Single device, multiple steps	Created Ready to Run Running Completed Failed/Errored	Total number of steps to be run (count of direct children)	Number of steps which have completed successfully (count of direct children where status is set to Completed)	Number of steps on which the operation did not start or errored (count of direct children where status is set to Unable to Run, Stopped, Failed, Canceled or Timed Out)
Multiple device, multiple steps	Created Ready to Run Running Completed Failed/Errored	Total number of devices on which sequence is to be run (count of direct children)	Number of devices which have completed successfully (count of direct children where status is set to Completed)	Number of devices on which the sequence did not start or errored (count of direct children where status is set to Unable to Run, Stopped, Failed, Canceled or Timed Out)

Implementation note: State is part of the database. The others may be calculated or cached by the WMI layer, depending on performance and other issues.

2.1.5.3.1 Per-Operation (Leaf-Node) Job State Definitions

The per-operation Job object represents a single process on a single target system. It can be in one of eight states. The states are:

State	Meaning
Created	The object has been created, but one or more of the other Jobs objects in the hierarchy have not yet been created
Ready To Run	The operation has been started, but is not yet running

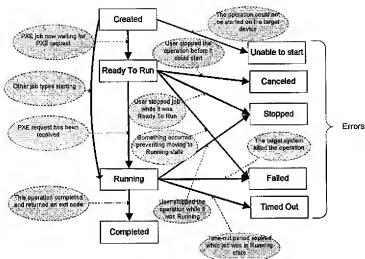
MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/finishing/specs/agmt01.asp>).

200955.DOC DOCUMENT IN 3027014-1 (NSCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/3/2003 1:04:00 PM 2/20/2003 12:32 PM 03/OCT/02 4:56 PM 04/SBR024

Running	The operation has been started on the target device
Completed	The operation has completed and returned an exit code to the controller
Canceled	The operation was canceled by the user before it could be started
Unable to Start	The operation could not be started on the target device
Failed	The operation was killed by the system while it was running
Stopped	The operation was stopped by the user while it was running
Timed Out	The controller did not receive a completion notification from the operation before the time-out period expired (note that the operation might actually have completed).

The valid transitions between the states are given in this diagram.



States are shown in black rectangles. The transitions are shown as thick arrowed lines. The causes of the transition are shown in the dashed ovals. If everything goes well, the states that the operation goes through are given on the left side (Created, Ready To Run, Running and Completed).

If something goes wrong, the operation will enter one of the four "error" states on the right-hand side.

The details of each state are given below.

- Created
New Jobs objects are assigned this state.
- Ready To Run

32

The controller is starting this job, but it is not yet running on the target device. This state only applies for PXE actions, this occurs until the PXE request has been received from the device.

- **Unable to Start**

If the controller was unable to send the job to the target, or the target sent a reply saying it could not start the job, the status is set to Unable to Start.

- **Running**

If the target responds that it started the process, the status for the job is set to Running.

Once the job is in this state, the controller collects output from the job. Output is of three types: standard output, standard error, and exit status.

- **Completed**

If the controller receives an exit status from the target device for this job, the status is set to Completed. This means that the process has exited successfully on the target. The job is now finished.

- **Failed**

If the process on the target device exits unexpectedly (for example, if it is killed by the system or a user), the status is set to Failed.

- **Stopped**

If, while the job is running, the user calls the method to stop the job, the controller will send a request to the target to stop the job. If the target returns that it killed that job, the status is set to Stopped.

Note: the controller does not set this status until the agent responds that the process has been killed. This is because the process might exit normally before the 'stop' request is received at the target – in which case, we should report the success of the process.

- **Canceled**

If the user stops a job that is in Ready To Run state, the status is set to Canceled. This is different from Stopped status, which means that the job did start running by was stopped before it completed.

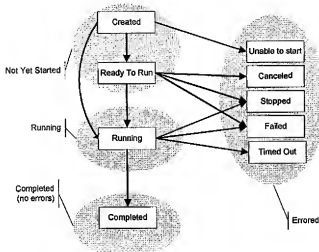
- **Timed Out**

Time-out period expired before job completed.

When we talk about rolling-up the status of operations into parent Jobs object, we are concerned with four "meta-states":

- Not Yet Started
- Running
- Completed
- Errored

This diagram shows how the meta-states relate to the states:

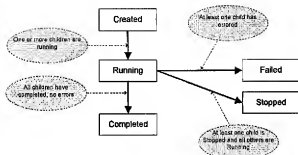


2.1.5.3.2 Parent State Definition

Parents can be in one of five states:

State	Meaning
Created	When Jobs object is first created
Running	When any one child is in the Running state
Completed	When every child is in the Completed state
Stopped	User stopped the job before it completed
Failed	When any one or more child is in Running state and one or more is in Errored state

The diagram below shows the state transitions for parent jobs.



The details of each state are given below.

- Created

New Jobs objects are assigned this state, until every job in the hierarchy has been created.

- Running

As soon as any one or more children move into the Running state, the state will be updated to Running.

The children that are running may move into the Completed, Stopped, Killed or Failed/Errors.

- Completed

All the children are in the Completed state.

- Stopped

Any one child is in the Stopped state (that is, it was stopped by the user) and all other children are Created or Running.

- Failed

If any one or more children enter an errored state (Canceled, Killed, Stopped, Timed Out or Failed/Errors), then the status is set to Failed.

If this Jobs object represents a task sequence, then this state means that the sequence did not complete. The "Children Completed" value can be used to determine what step of the process was last completed (if any).

If this Jobs object represents a job across multiple devices (either a tasks sequence on multiple devices, or an operation on multiple devices), this state means that one or more device either could not run the operation or sequence or the operation or

MICROSOFT CONFIDENTIAL © 2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/pressroom/pressroom1.asp>).

200955.DOC DOCUMENT IN 302701.1 - DISCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/2/2003 1:04:00 PM 2/2/2003 12:32 PM 03/OCT/02 4:55 PM 04/SEP/02

sequence failed while running (either killed by the system or stopped by the user). The "Children Completed" value contains the number of devices on which the sequence or operation completed successfully. The "Children Errors" value contains the number of children on which the sequence or operation failed to complete successfully.

Note that the operation or sequence may still be running on other devices. To check to see if any devices are still running the operation or sequence, the user must check the "Children Count", "Children Errors" and "Children Completed" values.

2.1.6 PXE Boot Actions and Default Sequences

This section only applies when ADS is being used in a PXE environment.

When a device does a PXE boot, what it is told to do depends on several settings on the controller. These settings are the PXEBootAction for this device, whether there is PXE job waiting for this device, or whether there is a per-device or system-wide job template to run.

The details of how the boot action is determined is given below for a given device doing a PXE boot. (This assumes that the PXE record has been matched to a specific device record – how this is done is defined in the Auto-discovery specification).

- 1) Check the setting of PXEBootAction setting for this device.
 - If this is set to 'Ignore', then the PXE server will never respond to this device. This takes this device out of control by the ADS system. END.
 - If this is set to 'Respond, move onto the next step.
- 2) Check to see if there is a job currently running against this device (CurrentJobID in the database). There are three possible situations:
 - There is a job currently running, and that job is a PXE job. In this case, this job determines the action to take. It can be one of boot to hard disk, boot to deployment agent, boot to virtual floppy or boot to third-party NBP. END.
 - There is a job currently running, but that job is not a PXE job (it is a program, script, namespace command for the target or imaging server, or controller program). This is an error. END.

Define behavior in this situation. Presumably log the error, and stop the sequence. Should it run the default job template? Probably not.

- There is no job currently running, in which case move onto the next step.
- 3) Check the value of the JobTemplate property of this device. This property contains the 'per-device default job template' to be run when a PXE request is received.
 - If the value is NULL or empty, move onto the next step.

- Otherwise, start the job specified in the job template (it may be a sequence or a simple job). If there is no job template with this name, log the error below (%1 is the device name, or if no name, "<unknown>" and %2 is the admin interface MAC address). END.

"The JobTemplate specified for device with name %1 and MAC address %2 does not exist. It has been ignored."

- 4) Check the value of the JobTemplate property of the Controller. This property contains the 'system default job template' to be run when a PXE request is received if there is no per-device job template exists.

- If the value is NULL or empty, move onto the next step.
- Otherwise, start the job specified in the job template (it may be a sequence or a simple job). If there is no job template with this name, log the error below (%1 is the device name, or if no name, "<unknown>" and %2 is the admin interface MAC address). END.

"The JobTemplate specified for device with name %1 and MAC address %2 does not exist. It has been ignored."

- 5) No PXE action is available for this device, so it will time out from the PXE stage and move onto the next step of its PXE boot order.

This is correct? Or do we hold it in bug.com for ever?

Should we log this? Will help with debugging what action is taken for a given device.

2.1.7 Task sequence Implementation

2.1.7.1 Examples

The examples below show in XML how each different combination would be expressed.

These are defined in the Remote Execution specification. If there is a conflict between these specifications, the Remote Execution specification should be used.

- Task Sequence

Not supported.

```
<task>
  <command-definition xmlns="/command>
    parameters/>
  </task>
```

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/learning/speclog/mkt1.asp>).

209955.DOC DOCUMENT IN 302791.1 - DISCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/3/2003 1:44:00 PM 2/3/2003 12:32 PM 03/OCT/02 4:55 PM 04/SEP/02 1

The `TargetType` and `Delivery` are always omitted. Internally, they default to `TargetType=Controller`, `Delivery=None`.

- Script or Program run on a Target Server

```
<task>
  <command>c:\Program Files\package.msi</command>
  <parameters>
    <parameter>ADDLOCAL="feature1"</parameter>
  </parameters>
</task>
```

Since `Delivery` is omitted, it is assumed to be "None" (that is, script or program is accessible to the device using the path given).

- Script delivered to over BMCP and run on a Target Server

```
<task delivery="bmcp">
  <command>c:\Program Files\package.msi</command>
  <parameters>
    <parameter>ADDLOCAL="feature1"</parameter>
  </parameters>
</task>
```

To specify that the script is downloaded over the BMCP (SSL HTTP) link, `Delivery` is set to "BMCP:bmcp".

If a script or program run on the target (either delivery mechanism) is known to do a reboot, the `doesReboot` attribute must be set:

```
<task delivery="bmcp" doesReboot="yes">
  <command>c:\Program Files\package.msi</command>
  <parameters>
    <parameter>ADDLOCAL="feature1"</parameter>
  </parameters>
</task>
```

- Script or Program run on the Controller

Do not know how to specify controller programs

```
<task>
  <command>runintarget="controller">
    c:\script\config_router.vbs
  </command>
  <parameters>
    <parameter>v1="c:\script\config_router.vbs"</parameter>
    <parameter>v2="c:\script\config_router.vbs"</parameter>
    <parameter>v3="c:\script\config_router.vbs"</parameter>
  </parameters>
</task>
```

`Delivery` is always omitted for Controller programs. `Target` must be set to 'controller'.

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/choice/sgp/choice/sgp1.asp>).

200955.DOC DOCUMENT IN 302701.1 - DISCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/2/2003 1:04:00 PM 1/3/2003 12:32 PM 1/3/2003 4:55 PM 1/4/2003 4:55 PM

- Native-mode Program run on a Target Server

```
<task doesReboot="true">
  <command>/bin/ncp/reboot</command>
</task>
```

Delivery is always omitted for native-mode programs and defaults to None. Note that the doesReboot attribute is set to true, since the reboot action will, by definition, cause the device to reboot. For the shutdown action (/bmonitor/shutdown), doesReboot should be set to false.

This is the same as a script or program run on a target device, except that the Command field contains a BMSS namespace instead of a file system or UNC path. The system can determine which it is by looking at the format of the Command field. If it starts with a slash, it is a native-mode namespace.

- PXE actions

General format:

```
<task>
  <command>/kxz/boot-ds/<command>
</task>
```

The TargetType target and dDelivery are always omitted.

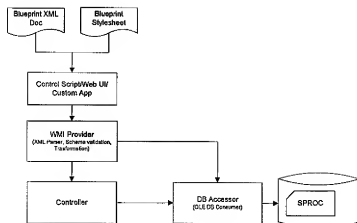
2.1.7.2 Task Sequence Architectural Description

WMI interface provides the top-level object model and the end users can access the object model through one of the following: Control scripts, Web UI and Custom applications that connects with WMI interface. The task sequence schema defines that the task sequence XML document can be either stand-alone or with references to other documents. So the task sequence author can produce either a single, self-contained document or a complex document that is a collection of multiple documents as modules. The complex document will be very useful for the following scenarios:

- To re-use the already existing task sequence modules that is designed to carry out a specific operation with a sequence of tasks. For example, Adding a user to a web server, Content replication on a file server, Creating a virtual root and configuring it, etc may be conceived as task sequence modules and the task sequence author can refer to these modules to achieve the end result through the use of either DTD entity referencing or XSLT.
- To produce the abstract task sequence with placeholders for the parameters. For example, task sequence author can produce a template task sequence document, parameter data document and a style sheet that defines how to merge the template with parameter document to produce a meaningful task sequence document.
- Mix of the above two scenarios.

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/learning/specs/legmt01.asp>).

200866.DOC\DOCUMENT IN 3027914-1-DISCLOSURE PACKET.DOC\TASK SEQUENCES-FUNCTIONAL-SPEC.DOC\LAST SAVED: 2/3/2003 1:04:00 PM/2/3/2003 12:32 PM/3/OCT/02 4:05 PM/04/S6P/024



After the transformation or resolving the entity references, the resulting document is parsed and validated with the task sequence schema. Then the validated task sequence document is further transformed into a simplified, flat XML document. In this flat document, both Blueprint and Task elements have similar structure and the Blueprint elements have additional attribute ID which is sequentially numbered in the document order. This transformation is required for the following reasons:

- Since Blueprint and Task elements are of Jobs Type (per DB schema), child elements info-set under the Task element is transformed into attribute-set on the Task element. So this transformation makes both Blueprint and Task element conforming to the same pattern.
- This architecture exploits the SQL Server's XML support. Querying the XML data stream with XPath predicate and converting the resulting info-set into the record-set are carried out with the use of Transact-SQL Command "OPENXML" as a part of the stored procedure. As per the estimated execution plan for the sproc execution, retrieving record-set from XML data stream with XPath predicate consumes 95% of the total estimated cost (CPU, I/O, etc...). Without this transformation, this step will be repeated more than once, thereby reduces the performance unnecessarily.
- The transformation includes an attribute ID to the Blueprint element that is sequentially numbered in the document order. This ID info is useful to generate the ID-JobID lookup table and this lookup table is used to fill in the ParentJobID field for the child jobs.

The transformed document is fed as an input to the stored procedure in the SQL Server that shreds the XML documents into Jobs and Jobinvocations table. Sproc is used instead of parsing and inserting the info-set into the Jobs and Jobinvocations tables as individual batch operations otherwise there will be a numerous batch operations like inserting a job record, inserting a jobinvocation record without duplication, retrieving the JobID and invocationID for the inserted Jobs and Jobinvocations records respectively, etc. These batch operations will

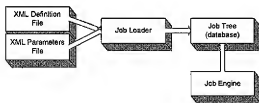
MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at
<http://www.microsoft.com/learning/aspnet/agmt01.asp>).

200906.DOC DOCUMENT IN 302701.1 - DISCLOSURE PACKET DOCTASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/23/2001 1:04:00 PM 02/23/2001 12:32 PM 03/02/02 4:55 PM 04/05/02 4

become a bottleneck if the SQL server and Controller software are deployed on different server.

2.1.7.3 Reading a Task Sequence

A task sequence is started by running the `ExecuteTaskSequence` method on either a `Devices` or `Sets` object. This method takes two external files: a sequence definition file and a sequence parameters file. The filenames of these two files are passed to the "job loader".



The following steps are performed by the job loader if the task sequence is to be executed on more than one target device:

- 1) Read the task sequence definition file and task sequence parameter file. If there are syntax errors in each file, the invocation of the task sequence fails. If either of the files does not exist, or is an invalid format, a WMI error is returned to the caller.
- 2) A "Jobs" object for this workflow is created. This is called the "multi-device sequence" Jobs object and represents the overall state of the workflow across all target servers. Give this Jobs object the next available JobID.
- 3) For each target device in the Set:
 - 4) Create a "per-device sequence" Jobs object for each target device as a child of the multi-device sequence Jobs object. Give each Jobs object the next available JobID.
 - 5) For each step in the task sequence definition:
 - 6) Replace any variables in the Parameters field with their values. If the definition file refers to a parameter that is not defined, return a WMI error and delete any Jobs objects created for this sequence.
 - 7) The step is either an operation (e.g. run a script) or another task sequence.
 - If the step is an operation: create a "per-operation" Jobs object for this step under the sequence Jobs object. Give each Jobs object the next available JobID.
 - If the step is a task sequence definition file: create a per-device sequence Jobs object for this task sequence, then start reading this task sequence file (effectively, this recurses to step 5 above. At the end of the included task sequence, go to the next step in the parent task sequence definition file).

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/permissions/permissions.asp>).

200955.DOC DOCUMENT IN 3027014 - DISCLOSURE PACKET.DOC TASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/3/2003 1:04:00 PM 02/03/2003 12:32 PM 02/02 4:56 PM 04/SEP/04

If the task sequence is to be run on a single device (instead of multiple devices) then there is no "multi-device sequence" Jobs object. Steps 2 and 3 of the above sequence are not performed, at the per-device sequence job created in step 4 has no parent Jobs object.

2.1.7.4 Implementation Notes

2.1.7.4.1 WMI interfaces

WMI interface exposes the following method under both target classes, Devices and Sets.

```
DWORD ExecuteBlueprint (
    [IN] string BlueprintPath,
    [IN] string StylesheetPath,
    ...
)
```

Arguments:

- BlueprintPath - file path of blueprint xml document.
- StylesheetPath - file path of style-sheet path. It is optional.

Return value:

- The RootJobID that represents the whole job sequence on the target.

2.1.7.4.2 Loading Task Sequence

Blueprint xml document is loaded and parsed with DOM or SAX interfaces. If the stylesheet is specified, the Blueprint xml document is transformed as per the stylesheet. Then the document is then validated with the Blueprint schema in the Schema cache. The parsing and processing of the Blueprint XML document is implemented using MSXML 4.0 as part of the WMI Provider. Blueprint schema is loaded into the Schema cache when the WMI Provider is loaded and initialized.

Execution of T-SQL batch commands and stored procedures are accomplished using OLE DB. They are implemented using OLE DB consumer templates in DB Accessor COM component. This component serves the DB accessing capabilities to both WMI provider as well as the Controller service. Serializing the Blueprint xml document into the Jobs and JobInvocations table (XML shredding) is implemented in the stored procedure.

After serializing the Blueprint xml document into the database, WMI provider invokes the ITaskManager::StartBlueprint method exposed by the Controller service. Blueprint Engine of the Controller service walks through the Jobs table executing the jobs sequentially till it either completes executing all jobs specified in the Blueprint or bails out if any of the jobs in the Blueprint encounters an error.

2.1.7.4.3 Executing Task Sequence

The job loader is part of the database interface (sactirdb). It has three functions, CreateJob, CreateBLJob and DeleteJob.

- JobID CreateJob(TargetType, TargetName, UserName, Delivery, CommandType, Command, Parameters, Description);

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/spec/agmt01.asp>).

200955.OCDOCUMENT-IN-3027014- DISCLOSURE PACKET DOCTASK SEQUENCES FUNCTIONAL SPEC DOCLAST SAVED: 2/3/2003 1:04:00 PM 2/3/2003 12:32 PM 03/OCT/02 4:05 PM 04/SEP/02 4

To create the single job

- JobID CreateBLJob(TargetType, TargetName, UserName, XML);

To create the jobs of task sequence.

- DeleteJob(JobID);

To delete the record of the job subtree from database.

2.1.7.5 Running a Task Sequence

- 1) Run the first unexecuted Jobs object (lowest JobID value) under each of the workflow parent objects. If this Jobs object is a workflow parent, run the first unexecuted Jobs object under this parent.
- 2) For each target server (i.e. each set of Jobs objects under the workflow parent object), wait for the previous job to complete. When it completes, go to step 7 if there are any more unexecuted Jobs objects under this workflow parent.

2.1.7.6 Implementation Notes

The job engine includes the job processor in sacitr.exe and the job object in sacitrb.dll. The engine will execute and update the job tree in database.

2.1.7.6.1 APIs

Three API s in the job engine:

- StartJob(JobID);

First, set the state of the job subtree from 'created' to 'ready to execute'.

Second, start to run the first job in the subtree; if the job command type is 'wait pxeboot', copy the command from JobInvocations table to devices table.

- CancelJob(JobID);

To set the state of the job subtree to 'canceled' if it has not started to execute.

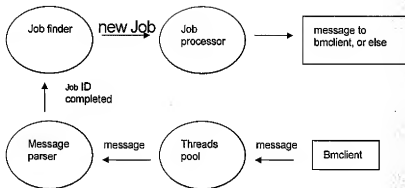
- StopJob(JobID);

To find and stop the running job in the job subtree and set its state to 'stopped'.

2.1.7.6.2 Workflows

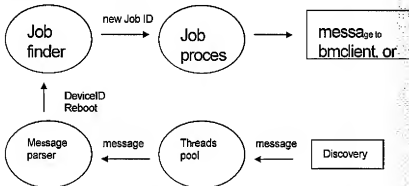
Four job workflows in the job engine:

2.1.7.6.2.1 Get job messages from bmcclient.



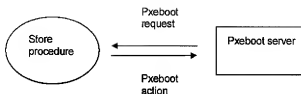
Job finder will find the next job id whose state is 'ready to execute' based on the id number from the completed job id.

2.1.7.6.2.2 Get reboot messages from discovery.



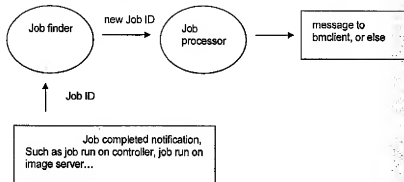
First, job finder will find the completed job id based on device id, its job type which is the 'wait rebooted', and its job state which is the 'running'. Second, job finder will find the next job id whose state is 'ready to execute' based on the id number from the completed job id.

2.1.7.6.2.3 Preboot



First, the store procedure will return the pxeboot action from Devices table to pxeboot server, and trigger another store procedure.
Second, store procedure will find the completed job id based on device id, its job type which is the 'wait rebooted', and its job state which is the 'running'.
Third, store procedure will find the next job id whose state is 'ready to execute' based on the id number from the completed job id.
Forth, store procedure will set the state of next job to the 'running'.
Fifth, if the next job is also the 'wait pxeboot', store procedure will copy the command in the Jobinvocations table to the Devices table.

2.1.7.6.2.4 Other job completed notifications



Job finder will find the next job id whose state is 'ready to execute' based on the id number from the completed job id.

2.2 UI DESCRIPTION

Task sequences are exposed to the user through the methods described in the Object Model specification. This specification contains no UI.

2.2.1 XML Schemas

See section 2.1.4

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/spec/agmt01.asp>).

200555.DOC DOCUMENT ID: 307701.4 - DISCLOSURE PACKET DOCTASK SEQUENCES FUNCTIONAL SPEC.DOC LAST SAVED: 2/3/2003 1:04:00 PM 02/02/2003 12:32 PM 03/OCT/03 4:66 PM 04/SEP/04

Exhibit 4

Windows

ADS Functional Overview

Microsoft Confidential. © 2001 Microsoft Corporation. All rights reserved.
FOR DISCLOSURE UNDER NDA ONLY

These materials are confidential to and maintained as a trade secret by Microsoft Corporation. Information in these materials is restricted to Microsoft authorized recipients only. Any use, distribution or public discussion of, and any feedback to, these materials is subject to the terms of the attached license. By providing any feedback on these materials to Microsoft, you agree to the terms of that license. If the license agreement has been removed, review the terms at <http://www.microsoft.com/forhire/properties/terms.asp> before using these materials.

Feature Information	
Feature Name	Automated Deployment Services
VP Technology Sponsor	Dave Thompson
VP Product Sponsor	Dave Thompson
Area	Scale-out
Related Features	Window Server
Related Documents	
Transit development (Y/N)	
Out of band release (Y/N)	Y
Which boxes does this feature ship on?	All () Home () Pro () Pro-64 () Blade () Server () Adv. Server () Adv. Server-64 () Datacenter () Datacenter-64 () Embedded ()
Requires Updates to:	SDK () DDK () Raskit () Online Product Docs and Help ()
Document Location	\\erra-sa\sd\public\specs\spec\big1.0
Spec Status	New
Document Security	Public (AllMS) (X) Private (Only Jim's group) ()
Contact Information	
PM Author	Paul Sutton
Dev Author	Ray Pedrize
Test Contact	Martin Holladay
Design	
Usability	
UA	Cynthia Nottingham
PSS	

Related Documents	
Functional Specification	See documents in \\erra-sa\sd\public\specs\spec\big1.0
Design Specification	See developer source depot (SASS)

Table of Contents

1	Platform Overview	5
1.1	ADS and the Microsoft scale-out vision	5
1.2	Feature Summary	5
2	Architectural Overview	7
2.1	Network Environments	7
2.1.1	PXE	7
2.1.2	Non-PXE	8
2.2	Architecture	8
2.2.1	Bare-Metal to O/S Architecture - PXE	10
2.2.2	Remote Task Execution	14
2.2.3	Task Sequences	15
2.2.4	Imaging	15
2.3	UI Description	16
2.4	API and Interfaces	16
3	Usage Scenarios	16
3.1	Deploying New Servers – Network PXE Boot	17
3.2	Deploying New Servers – Disk-based Pre-OS	17
3.3	Additional Scenarios anticipated	18
3.3.1	Installing Hotfixes and Service Packs	18
4	Scalability	18
4.1	Adding Image Servers	18
4.2	Adding NBS	18
4.3	Adding Controller	18
5	Upgrades	19
6	Terminology	19

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/learning/specs/term01.asp>).

200208.DOC DOCUMENT IN 302701.4 - DISCLOSURE PACKET.DOC ADS FUNCTIONAL OVERVIEW.DOC LAST SAVED: 2/3/2003 1:05:00 PM 2/3/2003 12:32 PM 12/19/2002 12:42 PM 05/OCT/02 1:34 PM 4

Functional Overview

1 Platform Overview

1.1 ADS AND THE MICROSOFT SCALE-OUT VISION

Over time the scale out services team will work separately, and in combination with other Microsoft and industry partners, to deliver a set of tools and a platform that significantly improve the way service providers and enterprise customers build, deploy and operate highly available and scalable services. Core to that platform will be the ability to rapidly purpose, administer and then re-purpose large numbers of windows servers in a datacenter environment.

As a first step towards addressing this platform need, Microsoft is adding an enhancement to the .NET server platform called ADS.

ADS is an enhancement to the .NET server platform that provides a programmatic way to automate the deployment and administration of large numbers of Windows 2000 and .NET Servers. As a platform, it can serve as the basis for an automated solution developed by a datacenter administrator or as an enabling technology that is key to an IHV or ISV deployment solution. The target end customer is a service provider or high end datacenter administrator who faces a unique set of challenges associated with deploying and maintaining 100s or 1000s of Windows Servers.

ADS provides data center administrators a programmatic way to remotely deploy operating systems to bare-metal servers, configure the systems (including use of DOS-only utilities) and enables a base level of management and maintenance of servers containing a full O/S.

IHV's and ISV's will now be able to leverage the low level functionality provided by ADS through a rich WMI interface and focus their resources on developing complete, intelligent, "out-of-the box" solutions that address the numerous deployment and administration needs of the broad Windows Server user community.

1.2 FEATURE SUMMARY

There are two generic operation capabilities that ADS will provide for Windows 2000 or Windows .NET servers in the data center environment:

- The ability to remotely purpose a system from bare metal to a useful state, or repurpose a system from one state to another state
- The ability to run extensible and configurable operations (scripts etc) on one or more systems from a single point of control and centrally collect those results

In addition to the core framework that is described below Microsoft will provide solutions to a select number of scenarios which may include:

- Purposing a system from bare metal to having an O/S running with basic parameters configured (name, etc)

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/comingsoonspec/agrm101.asp>).

200208.DOC DOCUMENT IN 2002014 - DISCLOSURE PACKET.DOC ADS FUNCTIONAL OVERVIEW.DOC LAST SAVED: 2/22/2003 1:05:00 PM 2/8/2003 12:32 PM 10/19/2002 12:42 PM 05/05/02 02:14:34 PM 5

- Querying systems and applying the appropriate hot fixes or O/S service packs to systems and images
- Repurposing existing system for a new function

To implement the above, the product is broken down into four major functionality blocks: imaging, secure deployment, task sequencing and remote script execution. Functionality of those blocks is as follows:

- **Imaging:** The ability to capture and manage Windows 2000 and Windows .NET Server images through standard Win32 tools, used either locally or remotely
- **Secure Deployment:** The ability to remotely boot a system, configure its BIOS, RAID and hard disks, and securely deploy a base O/S image
 - *Autodiscovery:* Ability to find booting servers on the network, and either treat them as a new system or map them to an existing system.
 - *Virtual Floppy:* Ability to configure BIOS, RAID and hard disks by running a self-contained DOS O/S (logical equivalent of putting a physical floppy into the system). Scriptable DOS RAID and BIOS tools must be obtained from the hardware vendor. These are not supplied as part of ADS. This feature requires that the system supports PXE.
 - *Network Boot:* Ability to remotely boot a system and determine whether it should run the deployment agent or boot into the O/S on hard disk. This feature requires that the system supports PXE.
 - *Non-PXE support:* Ability to store the management OS on a partition on the system, for situations where PXE is not supported in the production environment.
- **Task Sequencing:** The ability to define and administer a set of tasks to take a server through the deployment, imaging, and post-O/S configuration/administration steps required to build a complete server.
- **Remote Task Execution:** the ability to securely execute scripts or programs on a remote set of servers and centrally capture and record the results.

In addition to the above functionality blocks, there are a number of capabilities that cut across the entire product.

- **Rich Programmatic Interface:** As part of the overall ADS architecture a comprehensive WMI interface is provided. This enables all of the functionality to be programmatically driven from both script and compiled languages.
- **Setup:** the ability to install ADS
- **Samples/Scenario:** the samples we provide with the product to enable customers to get started fast addressing real problems they are experiencing
- **Documentation:** on how to use the product

- **Security:** apart from PXE and DHCP (which are standard protocols), all communication between servers under control of ADS and the ADS infrastructure servers is encrypted. Images can be encrypted.
- **Localization:** the initial version will be available in English only, although deployment of localized versions of Windows may be supported. Later releases may be available in additional languages.

ADS does not require Active Directory.

2 Architectural Overview

2.1 NETWORK ENVIRONMENTS

ADS supports both PXE and non-PXE environments. A PXE environment is one where the servers in the data center support PXE version 0.99d or higher. If the services support PXE, the data center administrator must make a decision about whether to use PXE, taking into account the security risks of the protocol. If PXE is used, it is recommended that the PXE NIC is connected to a management network that is kept more secure than the production network.

Advantages of using PXE include:

- Complete headless, remote administration of devices from bare metal to full OS
- Support for virtual floppy disks to allow for BIOS and RAID re-configuration
- Support for remote repartitioning and reformatting of hard disks

Primary reasons for not using PXE would be due to security considerations or because the data center servers do not support PXE functionality.

2.1.1 PXE

If PXE is used, the administrator must determine the size of the "DHCP domain". This is the scope of the network that receives a DHCP broadcast from another computer in that scope. By default, the DHCP domain is the same as a subnet or VLAN. However routers can be configured to forward DHCP broadcasts between subnets and VLANs.

Whatever size of DHCP domain is selected, a computer running the Network Boot Services (NBS) is required within each DHCP domain. The NBS must be connected to the network that is used for PXE broadcasts by the servers.

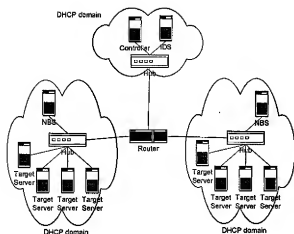
In addition to the Network Boot Services computer in each DHCP domain, a computer running the Controller service and Image Distribution service (IDS) must be deployed. The Controller must be able to communicate with the NBS in each DHCP domain, every device within each DHCP domain, and the Image Distribution service.

This diagram shows a typical network comprising multiple DHCP domains, showing the ADS infrastructure computers required. Note the Controller service and Image Distribution service may be co-located on the same computer. This diagram shows only the management network, not the production network (if they are different).

MICROSOFT CONFIDENTIAL. ©2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/spec/agmt01.asp>).

200228.DOC\DOCUMENT IN 30270-1-1-DISCLOSURE PACKET-DOCS\FUNCTIONAL-OVERVIEW.DOC\LAST SAVED: 2/3/2003 1:05:00 PM\2/3/2003-12-32 PM\1/19/2002-12-42 PM\26OCT02-4-34 PM 7



2.1.2 Non-PXE

If PXE is not going to be used, then the administrator can still use ADS to remotely deploy operating systems to servers, and administer them. However there are some limitations;

- In certain failure modes (e.g. hard disk failure) will cause the server to become unmanageable from the remote controller.
- Virtual floppy is not supported, so remote BIOS or RAID configuration is not possible
- A management OS must either be pre-installed into a partition of the hard disk, or installed into a partition just before being used (which might involve storing data previously in that partition, and later restoring it).
- The hard disk containing the management OS cannot be repartitioned
- Some disk space will be taken up by the management OS, reducing available disk space for the full OS or data

If these limitations are acceptable, then non-PXE environments can be supported. In ADS, non-PXE production environments are supported by installing a management OS onto the hard disk of the server using PXE. This is typically done in a staging area. Then the server is deployed to the data center. From this point forward it can be administered remotely from the controller, to allow the full OS to be downloaded, configured, boot, and then administered (subject to the limitations given above).

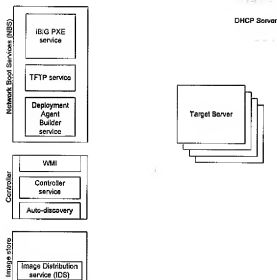
In this environment, the Network Boot Services do not need to be installed, so there is no need for this server in each domain DHCP.

2.2 ARCHITECTURE

The ADS system consists of four main parts, as shown in this diagram:

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/psoc/agmt01.asp>).

2003028.DOC DOCUMENT IN 3227511 - DISCLOSURE PACKET.DOC ADS FUNCTIONAL OVERVIEW.DOC LAST SAVED: 2/3/2003 1:05:00 PM 2/3/2003 12:32 PM 10/18/2003 12:42 PM 09/OCT/02 4:44 PM 8



For environments where PXE is being used, a computer running **Network Boot Services (NBS)** is required in each DHCP domain. The NBS enables a device to remotely boot up to the OS on disk, a virtual floppy, or to the deployment agent. In environments with a single DHCP domain, the NBS may be co-located same computer running the Controller service. Internally, the NBS comprises the ADS PXE service and the Deployment Agent Builder service. It also contains a TFTP service holding a library of network boot programs, virtual floppies disk images and auto-created deployment agent images.

The **Controller service** keeps a master record of computers in the data center, how they are arranged, what actions to take on next boot, and what operations can be performed on each target server. It acts as the control point for the ADS system and the target servers in the data center. The Controller service runs on a computer called the **Controller**. The Controller service uses a **database** to store its information. This database can be either an instance of MSDE located on the Controller, or an instance of SQL Server 2000 located locally or remotely. MSDE comes with ADS, while SQL needs to be licensed separately.

The computers within the data center that are to be managed from the controller are called **devices**. Each device can be controlled by at most one controller. Multiple controllers may be used in the environment, but they do not communicate between themselves, so the administrator is responsible for configuring each controller to know which devices it can manage.

The **Image Distribution service (IDS)** is used to store images that can be deployed onto the devices' hard disks. Each Controller service works with exactly one Image Distribution service. The IDS may be co-located with the Controller services on the Controller computer, but for performance reasons it is recommended that the IDS be located on a separate system.

The Controller service, Network Boot Services and Image Distribution service are collectively called the **ADS services**. They may be installed only onto computers running Windows .NET Server 2003 Enterprise Edition or Data Center Edition. ADS also includes additional

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/learning/specshg/m01.asp>).

20020800 DOCUMENT IN 302701-4—DISCLOSURE PACKET DGCADS-FUNCTIONAL-OVERVIEW.DGCLAST SAVED: 2/9/2003 1:05:00 PM 2/9/2003 12:32 PM 10/19/2002 12:12 PM 5/6/02 03:43:44 PM 9

In PXE environments, a DHCP service is also required. This is not part of the ADS feature. It may be on the same physical system as the NBS. A DHCP server is part of Windows .NET Server 2003 Enterprise Edition and Windows 2000 Advanced Server. A computer running the DHCP service is required within each DHCP domain.

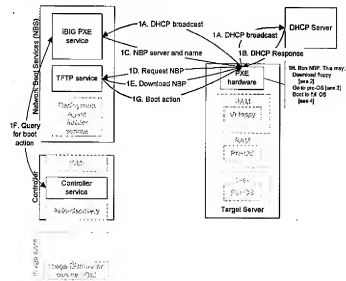
This section explains how these components work together.

This section describes how a target server goes from having no operating system installed ("bare-metal") to being fully configured and ready to perform useful work - a process called "purposing". It assumes that PXE is supported, and that the server has been placed into the data center, cabled and powered on.

Purposing a server typically consists of four stages. First is doing a PXE boot. The second is optionally loading and running one or more virtual floppies to perform DOS-level configuration. The third is booting into the deployment agent running in memory and downloading and configuring an OS final image onto disk. The fourth is booting into a final image (on the hard disk).

When a target server is booted, it must make itself known to the ADS PXE service (part of the Network Boot service or NBS). This is done using PXE.

The following diagram shows how the PXE boot process works. The labeled parts (1A) etc are referenced in the description that follows.



By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/specs/agmt01.asp>).

The target server boots using PXE, using a DHCPDISCOVER message request supplemented with an option identifying itself as a PXE client (1A). The packet contains the MAC address and SMBIOS UUID.

This is received by both the DHCP service and the ADS PXE Service. The DHCP service may respond with a broadcast DHCP OFFER message containing IP configuration for this target server (1B). Simultaneously, the ADS PXE service will process the discovery message. Using the MAC address in the discover message, it may decide to ignore the request, or respond to it. If it decides to respond, it will send an initial program called a network boot program (NBP) (1C to 1E). This may be an NBP defined by the user, or the ADS NBP, which is called `startnbs`. `startnbs` polls the PXE service for its next boot action.

During the polling process, the client returns to the PXE service and requests boot selection instructions. The PXE service opens a session with the ADS infrastructure and queries for boot action (1F).

The controller can tell it to do one of three things:

- Tell the target server to continue its BIOS boot sequence, that is, exit PXE
- Tell the target server to download and boot into the deployment agent
- Tell the target server to download and boot into a virtual floppy
- If no response is received within five minutes, the target computer is rebooted.

The first option is `boot-policy` is returned to direct the target computer to boot next BIOS boot device. Typically, this is the O/S image on the hard-disk.

The second and third options are described below.

A unique feature of the ADS PXE service is its flexibility to be configured to answer all broadcast requests, ignore all, or to be programmatically configured through the controller's object model specifying boot policy on a per device (target computer) ~~basis or set~~ (collection of devices or other sets).

2.2.1.2 Virtual Floppy Support

If the controller tells the ADS PXE service to download a virtual floppy, the ADS PXE service tells the `startnbs` NBP to download and run a virtual floppy. The name of the TFTP server and filename containing the virtual floppy image is supplied to the NBP by the NBS. The NBP requests the virtual floppy image from the TFTP server (2A and 2B). This is sent to the target server where it is stored in memory. The system then "boots" into the virtual floppy code, in DOS mode (2C).

The virtual floppy is an image of a bootable DOS floppy. It can contain DOS-based utilities that cannot be run within the deployment agent or target OS, and can be used to run BIOS upgrades, make BIOS configuration changes, or configure the RAID controller. ADS does not come with any tools to enable this, other than the virtual floppy support. The tools would come from the hardware vendor, and must be capable of being run from DOS without user interaction. The user is responsible for creating the virtual floppy image, including ensuring that the tools can be called with the correct parameters when the virtual floppy is "booted" (typically, by calling the tools from `AUTOEXEC.BAT`).

Once the virtual floppy tools have completed, the script run on the virtual floppy must do a reboot (2F). This will cause the server to start the boot process again with a PXE request -

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/pressrel/pressrel01.asp>).

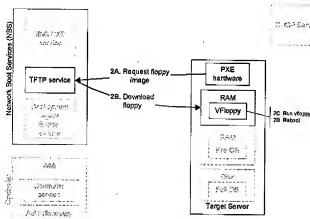
200928 DOCX DOCUMENT IN 302701.1 - DISCLOSE THE PACKET DOCAFS FUNCTIONAL OVERVIEW DOCLAST SAVED: 2/3/2003 1:05:00 PM 3/23/2003 12:32:54 PM 1/22/2002 12:42 PM 3/30/2002 4:34 PM 1

stage 1A. The controller state for the target server might have been updated to this time tell the target to either download and run a different virtual floppy, or to boot into the pre-OS.

Note that the controller has no way to know whether the virtual floppy completed successfully or not, since there is no communication from the DOS environment to the ADS controller. This is for security reasons, since the virtual floppy cannot authenticate itself to the controller. However the author of a virtual floppy script might make use of an unsecured share to communicate with the controller. The user would be responsible for implementing this, and for checking the results on the share from the controller.

The controller will maintain state information to determine what action to take on this boot, which might be more configuration steps, or it might be the downloading of a final OS image.

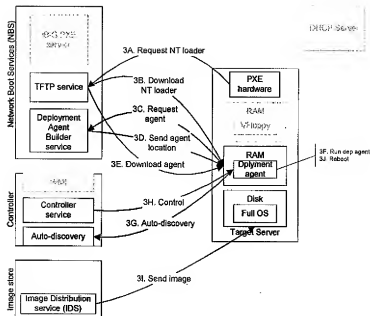
Comment [C11]: How does controller keep accurate state when system hangs and W/D fires or power is cycled and step does not complete successfully?



2.2.1.3 Deployment Agent Download and Run

If the controller tells the network boot server to download the deployment agent to the target server, the ADS PXE service tells the startnbs NBP to request the NT loader (3A) from the TFTP server (3B). This NT loader gathers some hardware information about the system and requests (3C) an OS image from the Deployment Agent Builder service. The deployment agent builder service dynamically builds a Deployment Agent and tells the loader where to obtain it from (3D). The loader then gets the agent and places it into the memory of the target server (3E). The deployment agent builder service caches a copy of this deployment agent instance for other target servers with the same hardware characteristics.

The deployment agent is started, running from memory on the target server (3F) and announces itself to the controller using autodiscovery (3G) and establishes secure communication with the controller. The controller can now send commands (3H) to the deployment agent to do things like partition the hard disks, format volumes, and download images (3I).



Downloading an image would consist of sending a job to the deployment agent. This will tell the deployment agent to run an image client utility. This utility requests an image from the Image Distribution service (IDS). The IDS contains a library of previous captured operating system images. The IDS sends the image from the target server, writing it to the partition specified in the command. After the image is downloaded, other jobs will typically be run to configure the OS (for example, to set the hostname and domain name).

When all the configuration tasks are complete for this invocation of the deployment agent, the deployment agent will be told to reboot the target server (3J). This will revert back to the initial PXE boot. Typically the controller will be updated so that on this boot, the target server is told to boot into the target (full) OS.

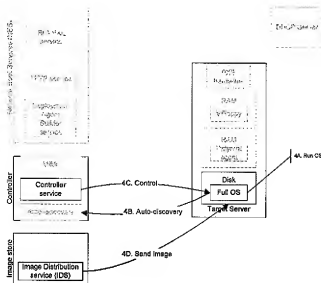
2.2.1.4 Target (Full) OS Boot

On this reboot, the target server again announces itself to the ADS PXE service. This again asks the controller what action to perform. If it allows the target server to continue to boot, the final OS will boot on the target server.

It boots into the OS on the hard disk (4A). If this OS contains the ADS Administration Agent service, it will tell the controller that it has booted, using autodiscovery (4B). The controller can now send additional remote execution tasks to the target server (4C), for example, to download additional images (4D), or run unattended installs, or to make configuration or provisioning changes.

The system is now purposed and ready for use in the data center. If the image contained utility agents (such as an SMS agent) it can now be managed remotely by SMS or other management applications.

If the system ever reboots, it will go back to stage 1. The Controller service can be configured to specify what action is to be taken if the system does reboot, such as booting back to the hard disk, or running the deployment agent.



2.2.2 Remote Task Execution

In order to work with hundreds or thousands of computers in a data center, the administrator applies a logical organization to the computers. There may be multiple organizations of computers for a single data center (for example, by customer or by function). The organization of computers is stored on the controller by placing the computers into sets. The administrator can work with sets of computers as if they were a single computer (to some degree).

The controller can tell one or more devices to perform an operation. The controller monitors the progress of the operation and stores the results. An operation to be run on a target OS can be a script, program, or a special operation (such as reboot). The system comes with a number of sample agent scripts to perform common typical configuration tasks, and the administrator can add additional scripts or programs as desired. The special operations are defined by the system and cannot be changed.

Remote tasks can be run on a target server when it is running the deployment agent, or when it is running a full OS which contains the administrative agent. In the deployment agent, the only tasks that can be run are the internal operations:

- Reboot the server
- Shutdown the server

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/spec/sgmt01.asp>).

200308.DOC DOCUMENT IN 3027014 - DISCLOSURE PACKET.DOC AGS FUNCTIONAL OVERVIEW.DOC LAST SAVED: 2/2/2003 1:05:00 PM 2/3/2003 12:32 PM (019/2002 12-42 PM) 6/6/2002 1:34 PM 1

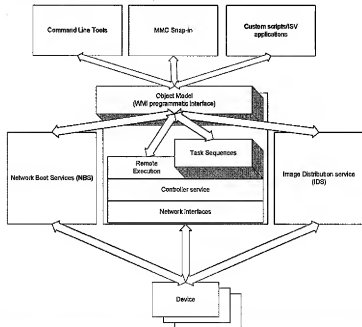
2.3 UI DESCRIPTION

ADS will have an MMC snap in UI that enables administrators to access a majority of the functionality described in this document including:

- Monitor devices/states
- Group and manage sets of devices
- Run jobs, task sequences
- View results of jobs, task sequences
- Manage imaging

2.4 API AND INTERFACES

This diagram shows a simplified view of how the Command Line Tools, Task Sequences and Object Model specifications relate. With a few exceptions, all functionality of the Controller service, Network Boot Services and Image Distribution service are exposed in the object model. The Command Line Tools (and the graphical user interface – not shown) are built on top of the object model.



3 Usage Scenarios

The following sections illustrate how some common server provisioning tasks might be implemented using ADS as the basis of a complete solution.

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.
By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/ipeds/agmt01.asp>).

200528 DOC DOCUMENT IN 202701-1- DISCLOSURE PACKET.DOC ADS FUNCTIONAL OVERVIEW.DOC LAST SAVED: 2/3/2003 1:05:00 PM 2/3/2003 12:33:PM 01/19/2002 12:42:PM 01/19/2002 4:34:PM 1-

3.1 DEPLOYING NEW SERVERS - NETWORK PXE BOOT

Assumptions: ADS controller and IDS installed. NBS and DHCP installed and configured. PXE boot can be supported either in hardware, or by keeping a PXE boot floppy in the device.

- 1) New server is powered on.
- 2) Server PXE boots. The administrator has already entered a record for this server on the controller, and associated it with a task sequence to be run when the server PXE boots. The task sequence is now invoked, and the following steps performed automatically.
- 3) A pre-created virtual floppy image is downloaded to the destination server and run, to configure the BIOS and RAID. At the end of the virtual floppy, the server reboots.
- 4) The deployment agent is downloaded to the destination server and started.
- 5) The deployment agent is told to repartition the disks, then to request an OS image from the IDS. Once the image is downloaded, the agent is told to personalize the sysprep.inf with the name, domain and static IP address of the server. The deployment agent is told to reboot.
- 6) The destination server boots into the O/S on the hard disk. This goes through mini-setup, and then reboots again.
- 7) The administrative agent on the destination server is told to run an unattended setup of a number of required hotfixes, then for an application required on this server.
- 8) Finally, the sequence emails the operator to say that the server is now up and running.

3.2 DEPLOYING NEW SERVERS - DISK-BASED PRE-OS

Assumptions: controller and IDS installed. DHCP or BOOTP servers are available.

Device has a boot partition that contains a pre-O/S which includes the ADS agent. This might be created in the lab when the device arrives, or might have been laid down by the vendor. It can be on the hard disk, or on a boot CD.

Comment [CN2]: Or might be on a floppy or CD.

- 1) New server is powered on.
- 2) Server boots into the pre-O/S and obtains an IP address through DHCP or BOOTP. The controller already has a record for this server, pre-created by the administrator, and associated with a task sequence to be run. This sequence contains the following steps.
- 3) An image is downloaded from an IDS server.
- 4) The name, domain and static IP address is set on the newly downloaded image
- 5) The destination server reboots.

- 6) The device boots into the O/S on the hard disk. This goes through mini-setup, and then reboots again.
- 7) The administrative agent on the destination server is told to run an unattended setup of a number of required hotfixes, then for an application required on this server.
- 8) Finally, the sequence emails the operator to say that the server is now up and running.

3.3 ADDITIONAL SCENARIOS ANTICIPATED

3.3.1 Installing Hotfixes and Service Packs

Administrator will be able to use the ADS scripting framework and a set of delivered scripts to automate the remote installation of hotfixes and service packs.

4 Scalability

In use, the administrator might determine that additional ADS capacity is required. This could be because existing servers are overloaded (e.g. too many PXE boot request for the NBS) or because the sizing limits are being exceeded (e.g. too many devices for one controller to support).

So the administrator might need to add additional IDS or NBS systems.

4.1 ADDING IMAGE SERVERS

If IDS systems become a bottleneck the administrator will want to add image servers. The administrator will need to install a second controller and associate the new image server to the controller.

4.2 ADDING NBS

An NBS is required per DHCP domain.

4.3 ADDING CONTROLLER

In this version, controllers do not know anything about other controllers on the network. Effectively they are independent of each other.

However the administrator could add additional controller capacity being logically partitioning the devices between two or more controllers, or creating a hierarchy of controllers.

■ Device Partitioning

The administrator would define some devices as being controlled by one controller, and other devices as being controlled by a different controller. This is purely a logical division, that the administrator would implement by the act of taking control of devices from a particular controller (they would also need to place the correct certificate on the device OS, which might require them to create an image per controller, or insert the certificate from the pre-OS phase.

Note that in pre-OS, any controller that has the correct certificate would be able to control the device during the pre-OS phase. This might be a different controller than the one that can control it in target OS, which would cause problems in task sequences, which run on a particular controller.

The NBS systems would have to be configured to forward boot action requests to the correct controller. This means that devices would have to be partitioned based on which NBS supports them. All the devices that boot via a particular NBS server would have to be controlled by the same controller. This is not typically a problem, since NBS systems usually support a smaller number of devices than a controller.

Note that if auto-creation of discovery records is enabled, all controllers will create records for all devices.

The user now has to go to the correct controller to perform operations. Operations on devices that are controlled by different controllers are more complex.

■ Ad-hoc Controller Hierarchy

The controller software does not understand a controller hierarchy. However the administrator could build an ad-hoc hierarchy. They would first start by partitioning the devices, as described above, into multiple controllers. Then they would create another controller, to be the "parent" of the individual controllers. They would also install the agent software on the individual controllers. Now scripts can be run from the parent controller against the individual controllers. These scripts could themselves invoke operations against one or more devices on the controllers.

5 Upgrades

The following upgrade scenarios are supported.

On the agent:

- Existing target server running W2K and the Add-on Pack agent, upgraded by uninstalling the SAK, upgrading to .NET server, then installing the ADS administration agent

On the controller:

- Controller running W2K, SAK 2.01 and Add-on Pack controller, by saving the database state to a file, uninstalling the AoP and SAK, then upgrading to .NET Server, then installing the ADS controller with the same database (which should make use of the existing database, upgrading it as necessary).

There is also controller migration:

- Migrate data from existing controller running W2K, SAK 2.01 and Add-on Pack controller to a new controller running .NET Server and ADS controller.

6 Terminology

Agent

Both Deployment Agent and Administration Agent

MICROSOFT CONFIDENTIAL. © 2001 Microsoft Corporation. All rights reserved.

By using or providing feedback on these materials, you agree to the attached license agreement (also available at <http://www.microsoft.com/licensing/terms/sgmt01.asp>).

200208.DOC DOCUMENT IN 327701-1-DISCLOSURE PACKAGE DOCS FUNCTIONAL OVERVIEW.DOC LAST SAVED: 2/3/2003 1:06:00 PM 02/03/2003 12:32 PM C:\92003\4212.PM\DOCT02-1-3-4.PM 1

BMon	[deprecated term] the basis for the Deployment Agent (see Deployment Agent)
BMon Builder	[deprecated term] replaced by Deployment Agent Builder service
BMSS	[deprecated term] part of the Deployment Agent and Administration Agent
Controller	The computer system that runs the controller service, and optionally also the Network Boot services and Image Distribution service.
Controller service	Software that implements controller functionality (including autodiscovery, BMSS, Win32 agent services, MSDXML, etc).
Deployment Agent	Software delivered over the network after a PXE boot to the memory of a target server and that then accepts operations from the controller (comprises BMon, BMSS and various native mode tools)
Deployment Agent Builder service	Service that creates a customized Deployment Agent based on the hardware on the Target Server
Device	One of the computer systems or other types of hardware in a datacenter. ADS version 1 only works with computer systems.
Production OS	An operating system installed for the purpose of doing useful work (for example, acting as a Web server or Exchange server). Contrast with a 'Management OS'. Sometimes called a 'Target OS'.
ADS PXE service	ADS's implementation of a PXE service
Image Distribution service	service that receives and sends image files
Image store	computer that runs the Image Distribution service
Job	The running of a job template on a single device or a set
Management OS	An operating system used to manage or service the computer system, such as by downloading a Full OS to the computer. The Management OS may be based on various operating systems (such as WinPE or .NET Server), but (for the purposes of ADS) must include the Administration Agent.
MDM	[deprecated term] see Multiple Device Management
Multiple Device Management	[deprecated term] remote execution of tasks on one or more target servers
Network Boot Services	combination of ADS PXE service and Deployment Agent Builder service
Operation	An action that is one of the following: script download, device program, controller program, device internal operation, controller internal operation
Pre-OS	often used to mean either the Deployment Agent or Management OS; sometimes used to mean specifically 'BMon' OS.
PXE service	see ADS PXE service
Set	A collection of one or more Devices
Target OS	see Full OS
Target Server	a computer system that can be managed by the ADS system
Task Sequence (or Sequence)	A sequence of steps, each of which can be an Operation or a Task
Full OS	See Production OS